МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: «Інформаційна технологія проектування комплексної системи захисту інформації будівельної компанії»

за спеціальністю 122 «Комп'ютерні науки»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студент групи ІТ.м-12 Лоцько Святослав Петрович

Кваліфікаційну роботу захищено на засіданні ЕК з оцінкою _____ «___» грудня 2022 р.

Науковий керівник _____ к.т.н., доц., Неня В.Г.
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2022

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В.о. зав. кафедри ІТ

_____ С. М. Ващенко

«___» _____ 2022 р.

ЗАВДАННЯ

на кваліфікаційну роботу магістра студентові

Лоцько Святослав Петрович
(прізвище, ім'я, по батькові)

1 Тема проекту *Інформаційна технологія проектування комплексної системи захисту інформації будівельної компанії*

затверджена наказом по університету від «04» листопада 2022 р. № 1013-VI

2 Термін здачі студентом закінченого проекту «12» грудня 2022 р.

3 Вхідні дані до проекту план робіт, література зі сфери захисту інформації, технічна документація системи Android, технічна документація Firebase, технічна документація SQLite

4 Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) інформаційний огляд предметної області, інструментарій та методології, формулювання інформаційної технології, моделювання додатку, реалізація додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) дерево критеріїв інформаційної безпеки, діаграма IDF0, діаграма варіантів використання, схема структури бази даних, опис таблиць БД, дизайн додатку, інтерфейси використаного програмного забезпечення, фрагменти коду та відповідні до них макети реалізації, діаграма Ганта, табличне представлення оцінки ризиків

6. Консультанти випускної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Інформаційний огляд	*Неня В.Г.*		
Інструментарій та методології	*Неня В.Г.*		
Моделювання інформаційної системи	*Неня В.Г.*		
Реалізація інформаційної системи	*Неня В.Г.*		

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Розробка концепту проекту;	03.10.22 – 22.10.22	
2	Пошук інформації	10.10.22 – 29.10.22	
3	Систематизація інформації	17.10.22 – 29.10.22	
4	Розроблення теоретичної частини роботи	24.10.22 – 31.10.22	
5	Формування функціональних вимог до практичної частини	17.10.22 – 12.11.22	
6	Створення діаграм IDF та варіантів використання	31.10.22 – 12.11.22	
7	Опис схеми роботи БД та її представлення в форматі ER діаграми	07.11.22 – 19.11.22	
8	Створення дизайну додатку	14.11.22 – 19.11.22	
9	Вибір засобів для технічної реалізації додатку	07.11.22 – 12.11.22	
10	Реалізація БД	31.10.22 – 05.11.22	
11	Розробка додатку	14.11.22 – 26.11.22	
12	Тестування додатку	21.11.22 – 03.12.22	

Магістрант _____ Лоцько С.П..

Керівник роботи _____ к.т.н., доц., Неня В.Г.

РЕФЕРАТ

Тема кваліфікаційної роботи магістра «Інформаційна технологія проектування комплексної системи захисту інформації будівельної компанії».

Пояснювальна записка складається зі вступу, 4 розділів, висновків, списку використаних джерел із 32 найменувань, 2 додатків. Загальний обсяг роботи – 205 сторінок, у тому числі 92 сторінки основного тексту, 4 сторінки списку використаних джерел, 109 сторінок додатків.

Кваліфікаційну роботу магістра присвячено розробці інформаційної технології проектування комплексної системи захисту інформації будівельної компанії. В роботі проведено аналіз систем комплексного захисту інформації, розглянуто інструменти та методи необхідні для захисту інформації, сформульована інформаційна технологія проектування комплесної системи захисту інформації. У роботі виконано моделювання мобільного додатку, на основі сформульованої інформаційної технології та його подальша розробка мобільного додатку. Результатом проведеної роботи є теоретична база, інформаційна технологія та мобільний додаток покриваючий частину функціоналу даної технології. Практичне значення роботи полягає у формулюванні інформаційної технології, яка поетапно описує формування базової системи захисту на підприємстві, що в свою чергу в парі з додатком, який реалізує частину представлених етапів, та теоретичною базою, яка надає інформацію для реалізації решти етапів, утворює просту в розумінні та практичну технологію для побудови базової системи захисту інформації.

Ключові слова: інформаційна безпека, інформаційна технологія, комплексна система захисту інформації, захист інформації, конфіденційність, цілісність, доступність, проектування, засоби інформаційної безпеки, android, SQLite, Firebase.

ЗМІСТ

ВСТУП

Інформація завжди була, є та буде неймовірно цінним ресурсом. В наш час інформаційні системи дуже щільно інтегруються у всі сфери людського життя, починаючи від закладів громадського харчування та закінчуючи великими компаніями зі світовими іменами, та навіть зараз не є можливим сказати що інформатизація досягла свого піку, вона швидше знаходиться на стадії розвитку.

Практично всі підприємства та державні структури використовують електронні системи для зберігання та передачі інформації. На даний момент схема роботи, яка була актуальна до цифрової ери, а саме робота виключно з фізичними носіями даних, є абсолютно нераціональною, а підприємства які її використовують, дуже швидко стають неконкурентоздатними. Навіть беручи до уваги всі переваги від використання інформаційних систем (пришвидшення роботи завдяки можливості швидкого обміну інформацією, відсутність необхідності в значній кількості фізичних носіїв та архівів для їх зберігання, можливість оптимізації роботи на основі даних якими оперує система і т. д.), не можна забувати про негативну сторону питання, а саме, електронний простір не є абсолютно безпечним. При роботі з ним також є певні проблеми, починаючи від технічних неполадок, що можуть привести до затримки в роботі підприємства, та закінчуючи зловмисниками, адже, як відомо, інформація в наш час вартує дорого, а там де є щось цінне, завжди є і ті, хто хоче отримати вигоду з нього незаконним шляхом.

Зловмисники є одночасно і фактором, який сприяє нестабільності при роботі з електронним простором, оскільки їх мотиви не завжди очевидні. Це може бути особа яка хоче отримати матеріальну вигоду шляхом продажу отриманої інформації, може бути колишній працівник, якого на його думку несправедливо звільнили та він хоче помститися, або просто хакер, який хоче довести самому собі, що він може обійти систему захисту. В той самий час протидія зловмисникам є доволі лінійною, якщо система безпеки налаштована відповідно до всіх стандартів

та вимог. Система захисту повинна бути універсальною, – їй байдуже яку ціль переслідує зловмисник.

Для забезпечення інформаційної безпеки підприємства необхідно використовувати комплексну систему захисту інформації. Така система – це сукупність всіх методів та засобів що використовуються для підвищення рівня захисту інформації. Те що називають комплексною системою захисту інформації (КСЗІ) насправді є поєднанням всіх засобів, які певним чином впливають на інформаційну безпеку підприємства. Важливо розуміти, що КСЗІ це не електронна система в звичному для всіх розумінні. Це є дещо більше. Це поєднання всіх чинників, які протидіють усім факторам, які можуть чинити загрозу для безпеки підприємства та додаткових засобів, які забезпечують працездатність, надійність тощо. Дана система не є тільки електронною, вона охоплює собою на однаковому рівні як антивірусні програми, так само і оф-лайн заходи для підвищення рівня інформаційної освіти серед персоналу. Той факт що вона працює не лише на інформаційному рівні, та організовує достатній рівень захисту для інформації компанії за допомогою технологічних засобів і програм, а також в цей момент увага уділяється також і фізичному рівню, робить дану систему захисту комплексною, оскільки у вас може бути відділена мережа, першокласні фахівці з інформаційної безпеки, дорого вартісні між мережеві екрани, інноваційне програмне забезпечення, але який в цьому сенс, якщо вашу серверну поставили біля кухні і на неї систематично проливають каву співробітники, або до офісу є вільний доступ не тільки у співробітників, а також і у кур’єрів зі служб доставки їжі, чи просто випадкових перехожих. На кожному комп’ютері в системі може стояти індивідуальний пароль, який гарантує що доступ до інформації на даному комп’ютері отримує лише ідентифікована особа, але в той самий час на стікері який приклеїли до рамки монітору написано цей самий пароль, що зводить на нуль його сенс.

Отже, саме комплексна система захисту інформації є неймовірно важливою складовою будь-якого підприємства, оскільки в наш час стрімкого розвитку

технологій, вона дозволяє забезпечити безпечну та стабільну роботу основних процесів підприємства, та захистити його як від внутрішніх, так і від зовнішніх загроз, при цьому беручи до уваги його специфіку та особливості.

1. ІНФОРМАЦІЙНИЙ ОГЛЯД

В даному розділі наводиться основна інформація щодо комплексних систем захисту інформації, що вони з себе представляють, на основі яких критеріїв будуються, якими стандартами регулюються та в загальному, чи потрібні вони, а якщо потрібні, то навіщо. Цей розділ повністю присвячений саме комплексним системам захисту інформації та відповідає на основні питання, які можуть виникнути при першому знайомстві з системами такого типу, а саме “Що?”, “Як?”, “Чому?” та “Навіщо?”. Тобто:

- Що таке комплексна система захисту інформації?
- Як працює та будується комплексна система захисту інформації?
- Чому вона працює та будується саме так?
- Навіщо вона потрібна?

1.1. Загальний огляд

Комплексна система захисту інформації(в подальшому КСЗІ) – це набір заходів фізичної та інформаційної безпеки, основною метою яких є мінімізація шансу несанкціонованого витоку даних за межі організації, а також забезпечення стабільної роботи основних систем підприємства, реалізація резервних схем роботи на випадок непередбачуваних обставин та підвищення рівню обізнаності співробітників щодо небезпеки нехтуванням захистом інформації.

Важливо розуміти, що незалежно від того, наскільки система є дорогою, продуманою або розробленою спеціалістами зі світовими іменами, вона не є ідеальною, та не зможе забезпечити 100 відсотковий захист, системи такого роду

призначені для того, щоб мінімізувати можливий рівень шкоди, а також підняти вартість доступу до інформації на такий рівень, коли зловмисник витрачає більше ніж отримує. Тобто, для кращого розуміння наводимо наступний приклад. Існує невелика компанія, її клієнтська база зберігається в Excel файлі на старенькому робочому ноутбуці. Ми розуміємо, що ця база представляє певну цінність, особливо для таких самих маленьких компаній, які працюють в цій самій сфері, але ця цінність не є значною. Тепер розглянемо два варіанти розвитку подій. Перший, якщо в цій компанії реалізована елементарна система захисту інформації, та другий, якщо нею знехтували. Ми не розглядаємо варіанти при яких в компанії реалізована гарна система захисту, є відділ який нею керує та слідкує, оскільки для маленької компанії це занадто нераціональні витрати, система захисту яку ми розглядаємо включає в себе мінімальну роботу з персоналом (тобто працівникам пояснили що не можна зберігати собі на флешки цю базу, для роботи з нею вдома), перенесення бази в хмарне сховище з доступом через пароль та реалізація системи корпоративних T-mail. Всі наведені засоби, не є дорого вартісними, по правді, це все є можливим організувати навіть безкоштовно, і не потрібно багато часу на реалізацію. В загальному все упирається в розмір бази, але в середньому за один робочий день, можливо навіть не цілий, цілком реально реалізувати такі засоби. При настільки невеликих укладеннях (декілька годин робочого часу), компанія отримує елементарну систему захисту, яка піднімає вартість доступу до інформації. Важливо також пам’ятати, що вартість самої інформації залишається однаковою, змінюється лише вартість доступу до неї [4]. Таким чином, в порівнянні цих двох варіантів, якщо з’являється зловмисник, ціль якого отримати цю базу та передати її конкурентам, то у другому варіанті він зможе отримати цю базу без особливих проблем: співробітники носять її на флешках, деколи забувають ці флешки на робочому столі, або дають друзям, відправляють її зі своїх персональних ноутбуків через незахищені мережі (наприклад мережа кав’ярні, чи кафе). Таким чином, отримати цю базу не є задачею для зловмисника, важче буде лише обрати спосіб, як її отримати, оскільки варіантів значна кількість. В першому

варіанті, коли є елементарна система захисту, насправді також зловмисник може отримати цю базу, але для цього йому доведеться провести більше ретельну та детальну роботу та, наприклад, застосувати свої навички з соціальної інженерії, щоб отримати пароль та доступ до корпоративного акаунту, далі підключитись через мережу підприємства і отримати цю базу. І в першому, і в другому варіанті, зловмисник отримує базу, але в першому варіанті, йому необхідно витрати набагато більше часу, сил та, можливо, фінансів. Ймовірніше за все ще до початку роботи він це зрозуміє і не буде навіть намагатись, оскільки втрачає більше ніж отримує [1].

Основними елементами КСЗІ є організаційні та інженерно-технічні заходи. Повертаючись до визначення КСЗІ є доцільним нагадати, що організаційні заходи забезпечують фізичну безпеку, а інженерно-технічні – інформаційну безпеку.

Організаційні заходи – заходи забезпечення інформаційної безпеки в межах певної організації, які описують концепт ІБ конкретно для специфіки та особливостей даної організації, а також включають в себе регламенти та інструкції щодо дій направлених на забезпечення ІБ підприємства [18]. Це такі як:

- інструкції для працівників в залежності від їх посад та обов'язків, щодо роботи з інформаційними системами (у випадках, коли їх обов'язки потребують такої роботи);
- регламент адміністрування елементів системи (тобто, яким чином ідентифікуються користувачі, як зберігається, змінюється та видаляється інформація, як проводиться робота з носіями інформації та їх знищення в випадку необхідності);
- організація заходів спрямованих на підвищення рівня інформаційної грамотності персоналу та навчанню їх правилам інформаційної безпеки;
- інструкції на випадок надзвичайних ситуацій (несанкціонований доступ до системи, поломка технічної складової системи).

Інженерно-технічні заходи – технічні заходи та методики їх використання спрямовані на забезпечення ІБ підприємства. Рівень інженерно-технічних заходів,

які будуть використовуватися на підприємстві напряму залежить від цінності інформації, яка буде захищатись, її типу, рівня критичності та вимог самого підприємства.

Інженерно-технічними засобами забезпечення інформаційної безпеки є:

- міжмережеві екрани;
- засоби шифрування даних;
- сегментування мережі;
- захищені канали для передачі даних;
- засоби захисту аудіо-інформації (засоби, які створюють технічний шум, та таким чином блокують роботу пристроїв для запису аудіо або його трансляції, зазвичай такі засоби встановлюються в окремих приміщеннях, які використовуються для обговорення стратегічних питань) [2].

1.2. Основні критерії інформаційної безпеки та їх вплив на КСЗІ

Основними критеріями для КСЗІ є ключові принципи інформаційної безпеки. Вони відомі як КЦД – конфіденційність, цілісність, доступність. Ці принципи є фундаментом в сфері захисту інформації, на них будується все, не залежно від масштабів організації, її специфіки, вимог і т. д.. Всюди, де заходить мова про захист інформації, невід'ємно слідують ці принципи. Конфіденційність, цілісність та доступність, одночасно є основними принципами інформаційної безпеки, критеріями оцінки КСЗІ та властивостями інформації. Розглянемо їх, виходячи з цих трьох позицій, для отримання розуміння того, як вони зв'язані між собою та чому вони є настільки важливими [6].

Конфіденційність – властивість інформації, яка дає можливість обмеження доступу до неї, тобто певна інформація є відома лише певному кругу осіб, та ніхто за межами цього кругу не має доступу до цієї інформації. Відповідно виходячи з

позиції конфіденційності, як принципу інформаційної безпеки, ми можемо сказати, що для забезпечення інформаційної безпеки підприємства є необхідним, щоб деяка інформація, позначена як конфіденційна, мала обмежений доступ до неї. Тобто вона не є загальнодоступною, а лише зареєстровані користувачі з правом доступу, мають можливість її використовувати. І з точки зору критерію оцінки КСЗІ, даний критерій оцінює відповідність реалізації цього обмеження доступу до вимог підприємства та інформації доступ до якої обмежується, і вже виходячи з цього робиться висновок, наскільки раціонально та доцільно був реалізований даний принцип.

Доступність – властивість інформації, яка гарантує, що користувач отримає доступ до необхідної йому інформації в той момент, коли йому це буде необхідно. Тобто, якщо користувач має можливість отримати доступ до інформації (цей доступ не обмежується конфіденційністю або посадою низького рівня в організації), то він отримає цю інформацію. Виходячи з позиції доступності, як принципу інформаційної безпеки, є можливим сказати, що, по-перше, даний принцип розповсюджується не тільки на працівників організації, а також на клієнтів у випадку, якщо є інформація, яку вони можуть отримати, наприклад, якщо організація займається виробництвом певної продукції та продажом її через інтернет-магазин. Тоді з одного боку доступність гарантує, що працівники компанії будуть отримувати доступ до ресурсів та даних, які потрібні їм для роботи, і неважливо якого роду ця робота: чи розробка нової лінійки продукції, чи запуск рекламної компанії, і в той самий час доступність гарантує те, що клієнти будуть мати можливість зробити замовлення в будь-який зручний для себе час та зі сторони організації з цим не буде проблем. Відповідно доступність, як критерій оцінювання КСЗІ, означає, що система є відмово стійкою на достатньому рівні, вона забезпечує стабільний доступ до всіх ресурсів, а також має резервні елементи у випадку виникнення проблем з поточними.

Проаналізуємо таку властивість як цілісність. Дана властивість інформації забезпечує, що інформація в системі є цілісною та не підлягала несанкціонованим

змінам, у випадку, якщо зміни були санкціонованими, то вже елемент даних після змін вважається цілісним. Таким чином, ця властивість дозволяє користувачам бути впевненими, що інформація, з якою вони працюють, є справжньою та до неї не було додано несанкціонованих змін, які її зіпсували. З точки зору цілісності, як принципу інформаційної безпеки, все доволі очевидно. Цей принцип гарантує, що інформація, внесена в систему, при подальшому використ анні є саме тією інформацією, яка вносилась раніше, та вона не підлягала змінам від третіх осіб. Відповідно оцінювання КСЗІ за цим критерієм відбувається в залежності від реалізації засобів, які забезпечують цілісність інформації. Зазвичай це засоби обмеження або розмежування доступу.

Виходячи з попередньо проведеного аналізу інформації щодо конфіденційності, доступності і цілісності, вже є можливим зробити висновок, що ці принципи є тісно пов’язаними між собою. По-перше, оскільки вони забезпечують інформаційну безпеку, то відсутність реалізації одного з них, або ж його реалізація на низькому рівні, ставить під загрозу всю систему. По-друге, багато засобів спрямовані на реалізацію декількох принципів водночас, наприклад сегментація мережі, одночасно дозволяє підвищити рівень конфіденційності, а також використовуватися для забезпечення цілісності та захисту інформації від несанкціонованих змін. По-третє, ці критерії мають свої підрівні, кожен з яких описує дві основні деталі [3]: наскільки буде реалізований захист та яким чином він буде реалізований. Також в них є додаткова властивість, це підрівні, які повинні бути наявними для реалізації поточного рівня. Більше детально ці підрівні буде проаналізовано в наступному підрозділі.

Таким чином ми маємо три основи на яких будується КСЗІ, а також ми знаємо що ці основи залежать одна від одної. Також варто пам’ятати, що хоч всі основні принципи є важливими та пов’язані між собою в рамках одного проекту, їх відносна важливість між собою буде відрізнятись в залежності від потреб підприємства та інформації, якою воно оперує. Наприклад, підприємство може оперувати інформацією, більшість з якої є конфіденційною. Таким чином, при

побудові КСЗІ варто звернути на це увагу та приділити конфіденційності максимально можливу суттєву увагу та ресурси, але при цьому також реалізувати заходи за рештою критеріїв на достатньому рівні.

1.3. Підвиди критеріїв та їх рівні

У попередньому підрозділі була наведено аналіз інформації щодо основних принципів захисту інформації, а також була згадано їх підвиди та рівні. У цьому підрозділі вони будуть проаналізовані більше детально.

Щодо конфіденційності в попередньому розділі було надано загальне визначення, яке в загальному можна спростити: конфіденційність захищає від несанкціонованого ознайомлення з інформацією та її розповсюдження. Це визначення описує суть, але воно є занадто узагальненим оскільки ми розуміємо, що конфіденційність захищає інформацію від несанкціонованого розголосу, але не розуміємо в яких саме ситуаціях, на яких рівнях, та яким чином це робиться. Тому розглянемо цей аспект більше поглиблено частково спираючись на результати дослідження [5].

Конфіденційність ділиться на 5 підтипів.

- Довірча – дана послуга дає користувачу можливість передавати інформацію від захищених об’єктів системи до інших користувачів, дана послуга має 4 рівні реалізації, ранжування яких відбувається випливаючи з глибини та детальності реалізації послуги.
- Адміністративна – дана послуга дає можливість керування інформацією від захищених об’єктів системи для її адміністратора, або спеціалізованого користувача з особливим рівнем доступу до неї. Ця послуга має 4 рівні реалізація, ранжування відбується випливаючи з тих самих факторів, що і в попередньому випадку.

- Повторне використання об’єктів – послуга, яка працює з повторним, або одночасним використання об’єкта системи, забезпечує те, що у випадку повторного використання об’єкта, цей об’єкт не буде містити інформації, яка залишилась після попереднього користувача або процесу. Ця послуга містить лише один рівень її реалізація, в якому прописані основні тезиси, що повинні бути реалізовані для відповідності цьому рівню, а саме, політика повторного використання об’єктів повинна бути однаковою для всіх об’єктів системи, перед наданням доступу до об’єкту новому користувача, спочатку відбувається повне анулювання прав доступу до цього об’єкта попереднього користувача, а також інформація внесена ним в об’єкт повинна стати недосяжною для нового користувача.
- Аналіз прихованих об’єктів – дана послуга реалізує контроль каналів передачі інформації, які не контролюються за допомогою інших послуг чи елементів КСЗІ. Дана послуга має 3 рівні, ранжування відбувається за діями, які відбуваються у відношенні до прихованих каналів, на першому рівні ці канали просто виявляються, тобто ми знаємо що вони є, вони є задокументовані, але нічого з ними не робиться, на другому рівні відбувається контроль цих каналів, ми знаємо що вони є, вони задокументовані, а також вони контролюються, а їх використання реєструється та задокументується, та останній рівень, це ліквідація, тобто ми знаємо що є приховані канали і просто їх ліквідуємо, відповідно логічним є що цей рівень забезпечує найвищий захист, оскільки при його реалізації підмножина прихованих каналів ліквідується та не чинить ніякої загрози для системи.
- Конфіденційність при обміні – дана послуга працює з забезпеченням конфіденційності інформації при її імпорті або експорті через незахищене середовище. Ця послуга має 4 рівні, які ранжуються відповідно від мінімальної конфіденційності при обміні до абсолютної конфіденційності при обміні, на останніх двох рівнях конфіденційність при передачі

інформації через незахищене середовище зберігається найкращим чином, зокрема через те, що на цьому рівні реалізується приймання запитів на обмін, не лише на основі атрибутів системи, а також атрибутів приймальника та відправника інформації, що дає можливість мінімізувати шанс втручання в процес третіх осіб.

Ситуація з цілісністю, така сама, як і з конфіденційністю. В попередньому розділі ми розглянули її загальну картину, а зараз виконаємо детальний аналіз.

Цілісність ділиться на 4 підтипи

- Довірча – дана послуга забезпечує можливість користувачів передавати інформацію від захищених об’єктів до інших користувачів, при цьому зберігаючи цю інформацію цілісною, тобто завдяки достатньому рівню реалізації цієї послуги, користувач, який отримує інформацію від іншого користувача системи, може бути впевнений, що це саме та інформація, яку йому надсилали і по дорозі до нього, вона не була змінена третіми особами. Ця послуга має 4 рівні реалізації, які ранжируються у відповідності до глибини захисту який реалізується.
- Адміністративна – дана послуга забезпечує можливість передачі інформації та керування потоками з даних для адміністратора системи, або користувача зі спеціальними правами в рамках цієї системи, ця послуга має 4 рівні реалізації, ранжирування яких відбувається випливаючи з рівнем захисту цілісності, який забезпечується послугою.
- Відновлення – послуга, яка забезпечує цілісність інформації, шляхом відновлення її стану до попереднього стану, процес відновлення відбувається через відміну операції, або сукупності операцій направлених на зміну об’єкта. Дана послуга має 2 рівні, це обмежене відновлення і повне відновлення. Відмінність полягає у тому, що при реалізації обмеженого відновлення, відміняється певна обрана множина операції по відношенню до об’єкту в певний проміжок часу, а в ситуації з повним

відновленням, проводиться відміна всіх операцій, які були виконані з об’єктом в певний проміжок часу.

- Цілісність при обміні – ця послуга реалізує засоби забезпечення збереження цілісності інформації при передачі її через незахищене середовище. Ця послуга має 3 рівні реалізації, та по аналогії з конфіденційністю, ця послуга на останньому рівні реалізації організовує запити при атрибутах зі сторони відправника та отримувача інформації, що в свою чергу забезпечує максимальний рівень захисту, тобто навіть при передачі через незахищене середовище отримувач може бути впевнений, що інформація яку він отримав не підлягала несанкціонованим змінам по дорозі до нього, а прийшла в тому самому вигляді, в якому її надіслав відправник. Для більшого розуміння, можна навести приклад використання елементів шифрування, коли атрибутом виступає ключ шифрування, при відправці інформації, вона шифрується відправником за допомогою певного ключа, таким чином по незахищеному середовищу передається зашифрована інформація, яка немає ніякої цінності та можливості до змін та лише коли вона доходить до одержувача, який також має ключ, за допомогою якого дешифрує її вона повертає свою цінність та значимість. Цей приклад також є справедливим для конфіденційності при обміні, оскільки засоби шифрування по перше не дають третім особам не маючим атрибута(ключа) прочитати інформацію, а по друге, зміни внесені в зашифровану інформацію, будуть відразу видні після дешифрації, таким чином реалізується і конфіденційність і цілісність інформації.

  Та останній принцип, доступність, він ділиться на 4 підтипи, кожен з яких забезпечує дотримання цього принципу в окремих умовах. Оскільки доступність відрізняється від конфіденційності та цілісності в основних своїх догмах, то і послуги якими вона реалізується відрізняються, як можна бачити з інформації про конфіденційність та цілісність наведеної

вище, в цих принципів є спільні послуги, такі як довірча, адміністративна, та при обміні, це зумовлено тим, що як конфіденційність так і цілісність направлені на збереження інформації, конфіденційність на збережені доступу до неї лише для авторизованих користувачів, а цілісність на збережені її незмінною. Тоді як принцип доступності, більше направлений на сам процес доступу, а не на інформацію. Тобто можна сказати що конфіденційність та цілісність працюють з інформацією, а доступність з тим щоб надати конкретно доступ до неї.

- Використання ресурсів – ця послуга дозволяє користувачу використовувати ресурси системи, тобто дає доступ до них. Ця послуга визначає реалізацію доступу до ресурсів для окремого користувача чи групи користувачів, визначає пріоритетність доступу та множини об’єктів до яких цей доступ може бути наданий. Дана послуга має три рівні реалізації.
- Стійкість до відмов – ця послуга реалізує можливість роботи системи при відмові її компонентів, тобто реалізація різних рівнів даної послуги дає можливість системі працювати навіть коли її елемент вийшов з ладу. Дана послуга має три рівні реалізації, перший рівень виділяється тим, що він не напрямлений на роботу з усіма елементами системи, а лише з обраною підмножиною, яка окремо регламентується. Цей рівень забезпечує роботу системи при відмові компонента з регламентованої підмножини. При цьому ймовірніше за все зі зниженням якості роботи до моменту усунення неполадки. Інші два рівні працюють охоплюючи всю систему зі всіма її елементами. Відмінність між ними полягає в тому, що другий рівень, як і перший, дає можливість системі працювати, але при цьому знижує якість цієї роботи до усунення неполадки, тоді як третій рівень є найвищим та навіть при відмові елементу системи дає її можливість працювати без зниження якості роботи. Практично при реалізації цього рівня лише

адміністратор буде знати про поломку, а рядові користувачі, навіть не помітять, що в системі є проблема.

- Гаряча заміна – ця послуга дає можливість адміністратору системи проводити заміну її елементів при цьому не перериваючи її роботу. Важливо також зауважити, що необхідність заміни може виникати, як через проблеми з елементом системи, наприклад його поломка, так і задля модернізації елементів системи, незалежно від обставин необхідності заміни. Адміністратор або технічний спеціаліст, повинен мати можливість зробити заміну, і при цьому не переривати роботу системи та зробити заміну таким чином, щоб після неї не довелось заново встановлювати програмне забезпечення. Послуга має 3 рівні реалізації. Перший рівень реалізує лише можливість модернізації системи, тобто він дає можливість виконати оновлення системи не впливаючи на його роботу, але при поломках елементів, будуть виникати проблеми і заміну провести без зупинки системи не вийде. Другий та третій рівні включають в себе перший, тобто дають можливість модифікації системи, а також дають можливість заміни проблемних елементів, різниця між ними в тому, що другий рівень не розрахований на всю систему, при його реалізації регламентується підмножина елементів, які можуть бути замінені без виникнення проблем в роботі системі, а третій рівень покриває всю систему і при його використанні є можливим замінити практично будь-який елемент не перериваючи роботу системи.
- Відновлення після збоїв – ця послуга дає можливість системі повернутись в робочий стан після збоїв. Послуга має три рівні: це ручне відновлення, автоматизоване відновлення та вибіркове відновлення. При реалізації першого рівня, в випадку, якщо в системі трапляється збій, який виводить її з ладу, адміністратор або технічний працівник, власними силами шукає проблему, її наслідки та усуває їх. Це рішення має право на існування завдяки своїй низькій складності та вартості, але при цьому воно зазвичай

займає багато часу. Другий рівень реалізації спрямований на те, що система сама проводить самоаналіз після збою, визначає проблему та усуває її. Це рішення є доволі швидким та надійним, але воно потребує великого обсягу попередньої роботи, зокрема з налаштування системи, визначення місць в ній з найбільшим шансом враження та підготування резервних елементів, які є підключені в систему в пасивному стані. Таким чином після виконання всіх цих дій у випадку збою система проводить самоаналіз, визначає, що проблема була в конкретному елементі, бачить що в неї є резервний елемент, який може покрити функціонал елементу, який був вражений та проводить відключення враженого елемента і підключає резервний. Також при автоматизованому відновленні адміністратор отримує звіт від системи, в якому вказано який елемент створив проблему, та як ця проблема була вирішена системою, і тоді адміністратор має достатньо часу щоб замінити чи виправити проблемний елемент, після чого переключити систему назад на нього. Для більшого розуміння можна навести приклад з генераторами електропостачання в будинках критичної інфраструктури. зазвичай вони підключені до мережі, але перебувають в пасивному стані до моменту коли стають необхідними. Якщо відбувається відключення електроенергії, то система електропостачання в будівлях критичної інфраструктури відключається від мережі, оскільки з неї не поступає енергія, та переключається на автономні генератори і працює на них до того моменту, поки в мережі знову не з’явиться електроенергія. Останній рівень реалізації схожий з попереднім, але відмінність полягає в тому, що він не потребує великого рівня попередньої підготовки та використовує наявні ресурси системи для продовження її функціонування з погіршеними характеристиками та при можливості повертає її в нормальний стан.

Узагальнений список всіх принципів та послуг та їх рівнів наведений на діаграмі нижче(рис. 1.3.1.). Даний рисунок наведений для узагальнення та візуалізації інформації наведеної в цьому підрозділі попередньо.

Рисунок 1.3.1. – Дерево критеріїв інформаційної безпеки та послуг, які забезпечують їх

1.4. Методи забезпечення інформаційної безпеки та їх види

Для забезпечення інформаційної безпеки підприємства при застосуванні комплексної системи захисту інформації використовують два типи засобів:

- організаційні;
- інженерно-технічні.

В даному підрозділі ми більш детально проаналізуємо ці типи, зокрема інженерно-технічні засоби.

Інженерно–технічні засоби використовують спеціалізоване програмне забезпечення та технології для того, щоб організувати інформаційну безпеку на підприємстві. Існує три основних види цих засобів:

- технічні;
- інженерні;
- криптографічні.

Якщо провести пошук в мережі щодо інженерно-технічних засобів захисту інформації, то є можливість знайти дуже багато різноманітних типів цих засобів. Складається відчуття що їх набагато більше ніж три, але насправді ці три види, які наведені вище, є основними. Решта різновидів тим чи іншим чином відноситься до одного з них. Часто буває так, що ці види не використовуються в чистому вигляді. Тобто існує певний засіб, який використовується на підприємстві, і його не можна віднести до одного з цих трьох видів. Це тому що він є змішаним, тобто він одночасно може бути віднесений до декількох типів. Також варто зауважити, що інженерно-технічні засоби більше відносяться до інформаційного захисту, тоді як організаційні до фізичного. Насправді в сфері інформаційної безпеки вже не існує чіткого розділення, а кордон між інформаційною та фізичною безпекою є дуже розмитий. Зокрема через багатофункціональність, яка стала одним з основних надбань розвитку технологій, і тоді, коли раніше компанії зосереджувались на тому щоб засіб робив одну задачу, але максимально якісно, і в наш час з розвитком технологій існує багато інструментів, які при додаванні в систему покривають відразу декілька напрямків і при цьому роблять це на достойному рівні, нічим не поступаючи інструментам, які чітко спеціалізуються на одному напрямі.

Розглянемо більш детально типи інженерно-технічних засобів, які наведені в рамках даної роботи.

Технічні засоби – зазвичай це програмне забезпечення, спрямоване на захист електронного простору від сторонніх факторів.

Інженерні засоби – обмежуючі конструкції, які ускладнюють доступ до критичних елементів системи, а також технічні засоби, які виконують задачу схожу з технічними засобами, але в фізичному просторі.

Криптографічні засоби – роблять інформацію нечитабельною для неавторизованих осіб, дають можливість вільно передавати інформацію, навіть в незахищеному середовищі, завдяки тому, що вона буде в зашифрованому вигляді, та не матиме абсолютно ніякої цінності без ключа, що дозволить її розшифрувати.

Також важливим фактором є те, що відповідно до законодавства України, а саме, закону України "Про захист інформації в інформаційно-телекомунікаційних системах", на юридичному рівні не існує розділення на технічні, інженерні та криптографічні засоби, оскільки в рамках законодавства всі засоби узагальнено називаються засобами технічного захисту інформації, і включають в себе всю множину інструментів. Комплексні системи захисту інформації повинні бути задокументованими, відповідати міжнародним стандартам та періодично проходити аудити сертифікованими органами [13], [15].

Щодо криптографічного захисту, ситуація складається таким чином, що криптографічні засоби захисту інформації по суті своїй були технічними, оскільки зазвичай криптографія це програмні засоби, які дозволяють шифрувати та дешифрувати данні використовуючи наявні алгоритми шифрування, а також генерувати ключі для цього шифрування. Але з розвитком технологій шифрування, як засіб захисту, відділилось від решти програмних інструментів через свою багатогранність, функціональність та корисність, і стало сприйматись як окремий вид засобів захисту інформації. Однією з особливостей даного виду захисту є те, що він працює не залежно від входження в систему, тобто, технічні засоби забезпечують безпеку системи, захищають її від зовнішніх загроз, та контролюють внутрішні, інженерні засоби працюють таким самим чином, але орієнтуються на фізичний простір системи, тоді як криптографічні заходи захищають інформацію, навіть якщо вона знаходиться за межами системи, в вільному незахищеному інформаційному просторі. Це є однією з причин, чому криптографічні засоби

вважаються окремим видом, а не підвидом технологічних. Як було сказано раніше, незалежно від того, як гарно влаштована система захисту, інформацію рано чи пізно доведеться передавати, і зазвичай цей процес буде проходити через незахищений інформаційний простір, де вона стане дуже вразливою, але в цьому випадку на допомогу приходить криптографія, яка забезпечить більш високий рівень захисту цієї інформації.

Тепер оперуючи інформацією наведеною в даному підрозділі, ми можемо вивести три твердження, які з неї випливають. Вони корелюють з результатами [8].

- Технічні, інженерні та криптографічні засоби мають одну мету (забезпечення інформаційної безпеки), але кардинально різні способи її досягнення
- З юридичної точки зору всі засоби підлягають під категорію “засоби технічного захисту інформації”.
- З технічної точки зору різні види захисту покривають різні області системи безпеки, тому спеціалісту, який займається проектуванням, або побудовою системи такого роду, необхідно розуміти відмінності між ними, їх особливості, переваги та недоліки в конкретних ситуаціях та на конкретних підприємствах.

1.5. Вплив КСЗІ на загальну діяльність підприємства

В загальному випадку вплив комплексної системи захисту інформації на діяльність підприємства повинен бути мінімальним з точки зору ускладнення робочих процесів чи додавання нових задач для персоналу, який ніяк не пов'язаний з інформаційними системами. КСЗІ та її введення до експлуатації на підприємстві несе за собою дві цілі:

- забезпечити захист інформації;

- оптимізувати робочий процес в інформаційному просторі.

З точки зору життєвого циклу, КСЗІ, не вважаючи на її комплексність та нелінійність, сприймається як стандартний програмний продукт. Життєвий цикл КСЗІ має 5 етапів[26]. Це наступні:

- Аналіз – проводиться аналітика організації, оцінка ризиків, визначення потреб та слабких місць, в загальному можна сказати що на цьому етапі аналізується підприємство та складаються вимоги, яким повинна відповідати КСЗІ таким чином, щоб покривати бізнес-процеси організації та забезпечувати достатній рівень захисту інформації [17].
- Проектування – виходячи з вимог визначених в ході аналізу, виконаного в попередньому етапі, розробляється проектний план, обираються конкретні засоби забезпечення інформаційної безпеки та розробляються технології їх впровадження.
- Розробка – безпосередня побудова КСЗІ встановлення програмного та технічного забезпечення, внесення необхідних інженерних рішень, а також розробка регламентів та інструкцій необхідних для організації захисту інформації.
- Тестування – проведення тестування готової системи, її робото здатності та дотримання вимогам встановленим в ході виконання першого етапу, зазвичай на цьому етапі з’являються нюанси, які не були враховані при проектуванні і з’являється необхідність вносити зміни в систему. В більш рідкісних випадках, можуть виникати критичні помилки, через невдале проектування, що призводить до необхідності повернення на один етап назад і повернення до розробки. У випадку з КСЗІ, шанс на такий варіант розвитку подій є мінімальним, окаільки система є комплексною, а її елементи зазвичай не впливають напряму на роботу одне одного, і шанс вчинити настільки критичну помилку при проектуванні, щоб некоректно працювала вся система є дуже малим.

- Впровадження та підтримка – останній етап. Систему впроваджують на підприємство та вона починає виконувати свої функції. Технічний спеціаліст або адміністратор системи займається тим, що слідкує за роботою системи, виправляє поломки у випадку їх утворення, вносить модифікації, та загалом підтримує систему у стані, в якому вона може адекватно виконувати свої функції [9].

Звернемо увагу, на вплив розробки системи на організацію, тобто, в залежності від масштабів організації та її ресурсного забезпечення є 3 варіанти розвитку подій

- Перший, організаціє є масштабною, має свій відділ технічних спеціалістів, а можливо і відділ з інформаційної безпеки, а також достатню кількість ресурсів для того, щоб виділити їх на розробку системи. В такому випадку саме працівники організації будуть займатись розробкою системи, її тестуванням, впровадженням та подальшим супроводом. Цей варіант є позитивним в плані якості виконаної роботи, якості комунікації виконавця і замовника, оскільки систему захисту інформації будуть розробляти люди, які вже працюють з інформаційною системою підприємства, знають в яких моментах зазвичай виникають проблеми та які загрози є найсуттєвішими. А з іншого боку, цей варіант, не зважаючи на якість продукту отриманого на виході, є дуже дорого вартісним та ситуативним. Тому цей варіант підходить для великих корпорацій або державних структур, які можуть довірити задачу своїм працівникам та надати фінансування на її виконання і бути впевненими, що робота буде виконана якісно в реаліях компанії.
- Другий варіант, це використання аутсорсингових компаній, або замовлення системи, у компанії, яка спеціалізується конкретно на роботі з КСЗІ. Таким чином компанія отримає систему розроблену спеціалістами, які мають певний досвід саме в сфері розробки комплексних систем захисту інформації. З мінусів даного варіанта є

можливим виокремити два пункти. Це, по-перше, можливі складності в комунікації, – класична проблема при роботі двох сторін, коли отриманий в результаті продукт в загальному є тим, що було замовлено, але в певних нюансах вийшло не так, як хотілось, і другий момент, це реалізація останнього етапу життєвого циклу, а саме підтримки, оскільки навіть після завершення розробки і введення системи в експлуатацію, буде необхідність наймати спеціаліста, який буде слідкувати за системою та лагодити її в випадках поломки.

- Третій варіант є найбільш бюджетним та простим, він підходить маленьким компаніям, які тільки починають свій шлях, їх сфера роботи може бути ніяк не пов’язана з інформаційними технологіями, але в час активної інформатизації, справу з ними мати доведеться в будь-якому випадку. Цей варіант полягає в найманні технічного спеціаліста, який буде займатись інформаційними системами, наприклад вести базу даних з клієнтами, надавати консультації в випадку виникнення проблем в роботі з ПК, а також введе елементарні заходи захисту інформації. Звісно, ці заходи будуть набагато простіші ніж в попередніх варіантах, та системою назвати їх також не є можливим, але як було сказано на початку даної роботи, задача КСЗІ не звести шанс втрати інформації до 0, а мінімізувати цей шанс та підняти вартість доступу до інформації таким чином, щоб зловмиснику це було не вигідно.

Отже вплив КСЗІ на підприємство буде мати лише позитивні наслідки, навіть не залежно від того, в якому форматі з представлених вище, була побудована ця система. Логічним є те, що більш якісна система забезпечить вищий рівень інформаційної безпеки підприємства, але в той же час, навіть елементарний набір засобів, який не буде коштувати організації практично нічого, значно підвищить рівень захисту в порівнянні з повною його відсутністю.

Також вартий уваги той факт, що введення в експлуатацію системи окрім явних наслідків, приведених вище, також підвищує рівень інформатизації

підприємства, що приводить до введення нових електронних систем, які пришвидшують роботу організації, а працівники, при цьому сприймають ці зміни більш просто, оскільки вони вже мали змогу працювати з подібного роду системою, і ще одна не сприймається для них, як щось незвідане.

1.6. Проміжний висновок

В ході проведеного аналізу розділу було розглянуто, що таке КСЗІ, з чого вона складається, яким чином оцінюється та взагалі, чому ця система є такою важливою та необхідною, незалежно від масштабу організації в якій вона вводиться. Основними тезисами виведеними в даному розділі є:

- КСЗІ – набір фізичних та технічних засобів, які забезпечують інформаційну безпеку підприємства
- Основними принципами інформаційної безпеки є конфіденційність, цілісність доступність. Вони ж є і критеріями, по яким оцінюють КСЗІ.
- Кожен з представлених принципів інформаційної безпеки має підрівні (послуги), за допомогою реалізації яких можна досягти певний рівень реалізації відповідного принципу.
- КСЗІ будується на основі потреб підприємства та складається з підмножини пов’язаних між собою елементів, які називаються засобами захисту інформації
- Засоби захисту інформації поділяються на технологічні, інженерні та криптографічні.
- Вплив КСЗІ на роботу підприємства повинен бути лише позитивним. Введена в експлуатацію система має окрім забезпечення захисту інформації оптимізувати роботу в електронному просторі та в той самий час не ускладнювати роботу працівників, а робити її лише простішою.

- КСЗІ не забезпечує 100% захист, вона лише реалізує захист інформації виходячи з наданих ресурсів та вимог, та таким чином підвищує вартість доступу до інформації і мінімізує шанс її втрати.

2. ІНСТРУМЕНТАРІЙ ТА МЕТОДОЛОГІЇ

Даний розділ присвячений безпосередньо методам захисту інформації та варіантам їх реалізації. В попередньому розділі, частково були висвітлені питання фізичної та інформаційної безпеки, в контексті КСЗІ та окремих її елементів. В даному розділі тема цих видів захисту буде розкрита більш детально з наведенням конкретних прикладів їх реалізації, а також ситуацій, коли захистом було нехтувано.

Незалежно від виду захисту (мається на увазі фізична або інформаційна безпека), який буде реалізовуватись першим, перший крок при побудові КСЗІ є однаковим, цим кроком виступає аналіз підприємства та зроблена на основі результатів цього аналізу, оцінка ризиків.

Оцінка ризиків — це процес, в ході якого визначаються основні вразливі точки організації, цінна інформація, яка може стати метою зловмисника, процеси, зупинка яких може привести підприємство до репутаційних або матеріальних втрат, та на основі отриманої інформації виділяється список можливих ризиків, втрат, які підприємство отримає при реалізації даного ризику, а також раціональних методів протидій цим ризикам [10].

2.1. Фізична безпека

Фізична безпека системи охороняє середовище, в якому ця система функціонує. Оскільки технологічний розвиток на даний момент, ще не знаходиться на тому етапі, коли всі дії проходять в інформаційному просторі, без необхідності в фізичних інструментах роботи з ними і люди не можуть транслювати інформацію

напряму в голову іншій людині, ми користуємось інструментами, які виконують ці задачі. Такими інструментами є будь-які носії інформації, це можуть бути мобільні засоби, персональні комп'ютери, сервера, тощо. Всі перераховані засоби підлягають під дві умови:

- вони містять інформацію;
- вони існують у фізичному світі.

Таким чином, необхідно забезпечити їх захист у фізичному просторі, оскільки не обов'язково бути хакером, чи добре знайомим з інформаційними системами, щоб ,наприклад, вкрасти флешку і отримати з неї данні. Фізична безпека повинна забезпечувати безперервний захист у просторі та часі. В той час, як інформаційна безпека забезпечує захист в нематеріальному, інформаційному просторі. Таким чином синергія цих двох видів захисту дозволяє забезпечити максимально повну безпеку інформації на підприємстві.

2.1.1. Методології забезпечення

Виявлено 5 основних напрямів фізичного захисту інформації:

- контроль фізичного доступу;
- контроль портативних приладів;
- захист при обміні інформацією;
- контроль інфраструктури;
- терміновий контроль.

Кожен з цих напрямів відповідає за окрему сферу фізичного захисту інформації.

Контроль фізичного доступу – цей напрямок організовує те, що інформація на її фізичних носіях, не потрапить до неавторизованих користувачів. Він характеризується обмеженням або розділенням доступу до певних секторів. Засоби реалізації даного напрямку захисту охоплюють попередньо визначений периметр підприємства та організовують засоби контролю на цьому периметрі. Основною метою даного напрямку є реалізація таких умов:

- контроль входу та виходу клієнтів та працівників організації, а також можливість фізичного обмеження входу та виходу за необхідності;
- ліквідація шансу на несанкціоноване/таємне проникнення на периметр організації;
- створення окремих секторів для робіт різного типу конфіденційності, або ж різного рівня доступу, можливо з окремими самостійними системами допуску в кожному з них;
- реалізація надійного режиму пропуску осіб на територію периметру і підтримання цієї системи.

При реалізації даного напрямку найчастіше використовується об'єктно-орієнтована методологія. Суть даної методології полягає в тому, що після визначення периметру, на якому повинен бути реалізований захист, проводиться його декомпозиція на об'єкти та зв'язки між ними, тобто, елементи на периметрі не сприймаються буквально (наприклад персональний комп'ютер керівника бухгалтерії), а сприймаються, як об'єкти з різним рівнем важливості. Такий підхід дозволяє створити максимально об'єктивну схему та побудувати співвідношення між елементами в ній. При цьому об'єкти, які на цій схемі виділяються, як найбільш критично важливі, в результаті отримують найбільшу увагу та найвищий рівень захисту, а завдяки прописаних взаємозв'язкам між об'єктами є можливим прослідкувати навіть ті місця в системі, які не є явно вразливими, тобто місця системи, об'єкти в яких не містять ніякої критично важливої інформації, і самі по собі не мають цінності, тому є логічним, що для них не має необхідності організовувати високий рівень захисту, але завдяки взаємозв'язкам, ми можемо побачити, що незважаючи на некритичність об'єктів, вони можуть бути напряму пов'язані з критичними об'єктами і таким чином, через них є можливо отримати доступ до інформації більш високого рівня [12].

Контроль портативних пристроїв – під портативними пристроями, маються на увазі носії інформації, які не є жорстко прив'язані до периметру, оскільки, наприклад, персональний робочий комп'ютер є розміщеним на периметрі

організації і захистом доступу до нього займається контроль фізичного доступу. Але що робити з носіями інформації, які не мають прив’язки до периметру, наприклад, флешки, ноутбуки та мобільні телефони. Насправді цей пункт є найбільш вразливим в системі фізичного засобу, оскільки ніхто не застрахований від викрадення пристрою або від того, щоб банально загубити десь флешку. Тому для того щоб хоч трохи забезпечити контроль портативних пристроїв залишається не так багато варіантів.

- Перший – мінімізувати використання портативних пристроїв для роботи з критично важливою інформацією з високими рівнями доступу. На великих підприємствах цей нюанс навіть прописаний в регламенті інформаційної безпеки.
- Другий – найбанальніший, бути уважним та, знаючи що на вашому пристрої є важливі дані, уділити йому максимум уваги або використати спеціальні криптографічні засоби для того, щоб навіть у випадку крадіжки, зловмиснику було важко отримати доступ до цих даних.

Захист при обміні інформацією. Цей напрямок реалізує захист у випадках, коли є необхідність передачі інформації. В загальному в плані фізичного аспекту захист при обміні реалізується в більшості своїй по території периметру, тобто у фізичному просторі. Цей напрямок означає внесення засобів, які дозволять ускладнити або повністю унеможливити для зловмисника перехоплення інформації. Одними з найбільш очевидних варіантів перехоплення даних є перехоплення пакетів, які передаються по мережі. Для цього зловмисник також має бути підключений до цієї мережі. Звісно, законним способом його ніхто не підключить, тому зазвичай зловмисник просто підключиться через лінію зв’язку. Але тут в дію вступає захист при обміні інформації, який може бути реалізований на фізичному рівні, наприклад, оболонкою кабелю зв'язку, яка подає сигнал при пошкодженні. Таким чином вирішується відразу два питання. По-перше, зловмисник не зможе підключитись непоміченим, оскільки для цього потрібен фізичний контакт з лінією зв’язку, а, по-друге, у випадку поломки лінії через інші

обставини, наприклад природного характеру, спеціалісти відразу будуть знати в чому проблема та навіть конкретно на якій ділянці ця проблема.

Також варто зауважити, що реалізація захисту при обміні інформацією є звісно добрим моментом, який підвисить рівень захисту системи. При цьому потрібно адекватно оцінювати ціну інформації, яка опрацьовується, і раціонально використовувати ресурси, виділені на організацію системи захисту. Наприклад, для комерційного відділу, який займається продажом продукції через мережу Інтернет, варто звернути більшу увагу на забезпечення цілісності та доступності ресурсу[30].

Контроль інфраструктури. Цей напрямок займається реалізацією засобів, які дозволять працювати інфраструктурним об'єктами. До них відноситься техніка, яка реалізує водопостачання, електрозабезпечення, теплопостачання. А задачами даного напрямку є захист обладнання, яке забезпечує ці послуги інфраструктури, а також реалізація можливостей по налагодженню цієї техніки у випадку поломки. Також цей напрямок включає в себе альтернативи, які дозволяють продовжити роботу підприємства, навіть в умовах, коли існують проблеми з інфраструктурними послугами не зі сторони підприємства. На даний момент цей напрямок є неймовірно актуальним, оскільки через руйнування енергосистеми України, підприємствам потрібно шукати альтернативні шляхи енергозабезпечення, для організації безперервної роботи організації. Такими шляхами на даний момент виступають генератори електроенергії, які дають можливість підприємствам працювати навіть в часи відсутності електропостачання, та мережеве обладнання Starlink яке використовується для організації зв'язку.

Протипожежний контроль. Даний напрямок фізичного захисту організовується та реалізується за підтримки пожежних інспекторів. Він включає в себе реалізацію засобів пожежної безпеки, які повинні бути прийняті та затверджені уповноваженим представником пожежної інспекції. Ці засоби дозволяють захистити підприємство від можливих пожеж, а у випадку їх виникнення реалізовувати розроблений та перевірений план евакуації, засоби, які

в автоматичному режимі почнуть боротьбу з пожежею, та дозволять залишити ситуацію контрольованою до моменту прибуття пожежників спеціалістів [16][28].

2.1.2. Інструменти

Розберемо більш детально інструментарій, який застосовується для реалізації фізичного рівня захисту інформації.

Для реалізації контролю фізичного доступу застосовуються такі інструменти, як:

- охоронна система;
- камери нагляду;
- двері з замками;
- перегородки для розмежування фізичного доступу;
- датчики руху;
- сигналізація;
- спеціалізовані автоматичні системи авторизації.

Для контролю портативних пристроїв не має певних фізичних засобів реалізації, оскільки найбільш небезпечна зона для портативних пристроїв знаходиться за межами периметру, на якому реалізується система захисту. Таким чином, за межами цього периметру система не має ніякої дієвості і єдиний захід, який може допомогти, це проведення семінару для працівників, на якому буде детально роз’яснюватись, чому не варто носити критично важливу інформацію на портативних пристроях, а також чому краще взагалі не зберігати робочу інформацію на особистих портативних пристроях.

Інструментарієм для реалізації захисту при обміні інформацією є наступні.

- Оболонка для ліній зв’язку з виявленням пошкодження.
- Засоби генеруючі шуми, для нівелювання шансу підслуховування аудіо інформації. Зазвичай засоби такого формату встановлюються не по всьому периметру, а в окремо виділеному приміщенні, яке в подальшому

використовується для обговорення важливих питань без ризику того, що інформація може бути вкрадена.

- Засоби криптографії. Насправді цей пункт більше відноситься до інформаційного захисту, але коли йдеться про захист при передачі інформації через незахищений простір, не можна забувати про криптографічні засоби.

Засоби контролю інфраструктури поділяються на два підпункти, це конкретно засоби контролю існуючої інфраструктури, а також засоби забезпечення альтернативних джерел при несправності основних. До першого підпункту відносяться такі інструменти як:

- наявність резервних запчастин;
- засоби контролю цілісності обладнання;
- періодичні діагностики;
- детектори несправності;
- наявність спеціалістів, які зможуть полагодити поламку у випадку виникнення проблеми;
- наявність спеціалізованих інструментів для виконання ремонтних робіт.

Та до другого підпункту відносяться:

- генератори, які дозволяють забезпечити безперебійне електропостачання до організації;
- мережеві станції, для організації зв’язку;
- налаштована локальна мережа.

Пожежний контроль є критично важливим і необхідним. Насправді навіть з юридичної точки зору необхідно забезпечити цей захист вже навіть не говорячи про те, що небезпека, від якої він захищає, становить ризик для здоров’я та життя працівників. Інструментами необхідними для реалізації цього захисту є

- Пожежна сигналізація;
- Ручні засоби пожежогасіння;
- Автоматизовані системи пожежогасіння;

- Розроблений попередньо план евакуації співробітників;
- Детектори диму та жару.

2.2. Інформаційна безпека

Основною метою інформаційної безпеки на підприємстві є забезпечення достатнього рівню захисту використовуючи технічні засоби для цього. В попередньому підрозділі мова йшла про фізичну безпеку, яка забезпечувала фізичний захист об’єктів на підприємстві. У випадку з інформаційною безпекою, вона не є зримою у порівнянні з іншими видами небезпек. Практично будь-який інструмент фізичного захисту є матеріальним. Взяти до прикладу датчики руху. Людина бачить, що ось є датчик, він працює і потрібний для того, щоб захистити підприємство від несанкціонованого проникнення. У випадку з інформаційною безпекою, дії по захисту переходять в інформаційний простір, а засоби, які забезпечують захист, також не є матеріальними. Це програмні продукти, які виконують свою задачу. Для кращого розуміння наведемо дві практично аналогічні ситуації в різних просторах, з одного боку, зловмисник зайшов в офіс та вкрав фізичний носій з базою даних, а з іншого, зловмисник зайшов в систему та вкрав ту саму базу даних з неї. В першому випадку, для протидії використовуються засоби фізичного захисту, наприклад охоронець, який бачить людину, яка не є працівником організації та не пропускає на територію організації, або якщо зловмисник вже потрапив на цю територію, виводить його з неї. В другому випадку, сценарій дій абсолютно ідентичний, але в ролі охоронця виступає програмний засіб, наприклад між мережевий екран, який бачить, що в систему намагається проникнути неавторизований користувач, або ж з мережі підприємства йде передача підозрілого трафіку в незахищений простір, в такому випадку, якщо система автоматизована, вона може сама перекрити передачу цього

трафіку, а якщо не автоматизована, то наддасть сигнал технічному фахівцю, чи адміністратору мережі, Який вже сам прийме рішення щодо алгоритму дій в цьому випадку.

Отже, виходячи з наведеного вище прикладу, є можливим сказати, що по суті своїй обидва типи захисту є дуже схожими між собою, і відмінність лише в тому просторі, де вони виконують свої дії, і відповідно в орієнтації інструментарію. В деяких ситуаціях ці два види захисту тісно переплітаються та виконують суміжні задачі. Тому варто пам’ятати, що в комплексній системі захисту інформації необхідна реалізація на необхідному рівні обидвох типів захисту, оскільки недостатній, або неякісно зроблений захист на одному з них, ставить під загрозу систему в цілому.

2.2.1. Методології забезпечення

Розглянемо три основні напрямки безпеки, які забезпечуються технічними засобами на підприємстві:

- захист інформації при її передачі за межі інформаційної системи підприємства;
- захист працівників від них самих;
- розмежування доступу до інформації в системі.

Захист інформації при її передачі за межі інформаційної системи підприємства – цей напрямок забезпечує те, що, по-перше, передача інформації за межі інформаційної системи підприємства може бути лише авторизованою, тобто вона буде дозволеною, а також є чітка прив’язка хто саме, яку інформацію та кому передає. По-друге, засоби даного напрямку реалізують безпеку інформації за межами інформаційної системи, тобто вони мінімізують шанс того, що інформація при передачі буде перехоплена та прочитана третіми особами.

Захист працівників від них самих – це напрям безпеки, який організовує обмеження доступу співробітників до сумнівних ресурсів, а також автоматичне блокування системою таких ресурсів у випадку, якщо система бачить ризик для себе. Як було сказано раніше, фізична безпека часто переплітається з

інформаційною та доповнює її. В конкретному випадку можуть також застосовуватись організаційні заходи, на яких до працівників буде донесено основи інформаційної гігієни та надано поради щодо її дотримання.

Розмежування доступу до інформації в системі – даний напрям реалізується завдяки сегментуванню системи на сектори та організації ієрархічної структури доступу до них. Метою даного напрямку є те, що до інформації отримують доступ лише ті працівники, які мають на це відповідні дозволи. Відповідно ієрархічна структура будується таким чином, щоб витримати дві основні вимоги:

- особи, яким для виконання своїх робочих обов'язків необхідний доступ до певної інформації повинні мати до неї доступ;
- особи, яким ця інформація не потрібна для виконання робочих завдань, цього доступу мати не повинні.

Існує дві основні методики розмежування доступу суб'єктів інформаційної системи до її об'єктів, це дискреційна методика та мандатна. Відповідно суб'єктом системи вважаються користувачі, а об'єктами саме інформація.

Дискреційна методика полягає в тому, що права доступу суб'єктів до об'єктів регулюються списками доступу або матрицею доступу. Існує декілька варіантів використання цієї методики:

- Конкретний об'єкт (тобто інформація) має певного власника (тобто користувача системи) і цей власник сам вирішує кому надавати доступ до об'єкту, а кому ні;
- В системі існує адміністратор, чи суперкористувач, який має максимальний рівень доступу до всіх об'єктів, та саме він вирішує, яким чином побудувати розмежування, кому доступ надати, а кому ні;
- Користувач системи, яки має певне право доступу має можливість передавати його за своїм бажанням, чи по необхідності іншому користувачу.

Також ці варіанти побудови системи доволі часто зустрічаються в змішаному вигляді, тобто в системі може бути і суперкористувач, який призначає права доступу, а також користувачі мають право передавати свої права один одному.

Мандатна методика є більш орієнтованою на об'єкти системи, тобто, тоді коли дискреційна прив'язується до прав доступу, які має користувач, і з цими правами він має можливість звертатись до об'єктів, мандатна працює по схожій схемі, але навпаки. Всі об'єкти системи поділяються за рівнем конфіденційності, кожному з них присвоюється певна група, а користувачі в свою чергу, мають право доступу до об'єктів лише з конкретної групи конфіденційності та нижче. Таким чином права доступу будуються виходячи з того, чи рівень доступу суб'єкта є достатнім для об'єкта.

Для кращого розуміння, можна уявити систему координат, на якій позначені всі користувачі та об'єкти системи, відповідно до осі Y, чим вище позначка, тим вищий рівень доступу у користувача та рівень конфіденційності в об'єкту. Якщо за доступом до об'єкту звертається користувач, який на цій системі знаходиться вище за об'єкт, він отримує доступ, а якщо користувач нижче за об'єкт, це означає, що його прав недостатньо.

Мандатна методологія обмеження доступу зазвичай використовується в системах з високим рівнем важливості, там де є можливим розділити об'єкти відповідно до їх конфіденційності. Наприклад, така система використовується в державних структурах до документації, яка позначена різними рівнями грифів, таким чином об'єкти по суті своїй вже є поділеними, а мандатний метод використовувався у фізичному просторі, тому доволі просто та доцільно перенести його в інформаційний простір [11].

Також існує модель, яка використовується для виявлення слабких місць в системі організації, вже після того, як в неї внесена комплексна система захисту інформації. Дана модель називається модель порушника, вона використовується для тестування системи та прогнозування можливих загроз. Ця модель представляє собою симуляцію зловмисника, особи яка випадково, чи з поганим умислом

отримала доступ до системи. Сама модель, це по суті створення різних типажів порушників та накладення їх дій на поточну систему. Наприклад, спеціалісти з безпеки припускають тип порушника, тобто чи він потрапив в систему зовні, чи він є працівником компанії та має певний рівень доступу, чи він потрапив в систему випадково, чи заради матеріального збагачення вчинив це умисно, робляться припущення про рівень навичок порушника і так далі. Загалом можна сказати, що на основі свого досвіду, спеціалісти з кібербезпеки створюють окрему особистість, яка несе загрозу для системи, моделюють її кроки виходячи з умови того, як би вона вчинила для досягнення поставленої мети. Загалом використання моделі порушника не є чітко формалізованим методом, але використовується доволі часто на практиці, оскільки при побудові системи дуже часто спеціалісти упускають деталі, і лише погляд на цю систему з іншого боку, з боку особи, яка має на меті не організувати захист, а навпаки його зруйнувати, освітлює багато слабких місць, на які при побудові увагу не звернули. З цією самою метою великі компанії користуються послугами білих хакерів, які за грошову винагороду, а деякі заради інтересу, проникають через системи захисту корпорацій та потім вказують в яких саме місцях вони побачили вразливість, яким чином її використали, та як краще за все цю вразливість перекрити [20].

2.2.2. Інструменти

Інструменти технічного захисту для працівників, які захищають працівників від них самих, це листи обмеження доступу та антивірусні програми. Вони зазвичай є підключені до доволі масивних баз з інформацією про шкідливе програмне забезпечення, які постійно оновлюються. Таким чином працівники будуть в безпеці як мінімум від найбільш актуальних загроз, які можуть виникнути через необачне користування мережею Інтернет. Щодо листів обмеження доступу, варто зауважити, попередньо в даному розділі було описано методи, які дозволяють розмежувати доступ, і в цьому пункті мова йде не про них. Листи, описані в даному розділі, це по суті список доменів, доступ до яких є заблокованим в рамках робочої мережі. Зазвичай, вони не розмежуються по користувачам, а є

спільними для всієї організації. Суть роботи таких листів проста, при запиті до веб ресурсу, цей запит проходить через наприклад роутер, система якого перед тим, як надіслати запит до ресурсу, перевіряє, чи є адреса ресурсу до якого звертаються в списку, якщо є, то запит навіть не проходить далі, а користувач відразу отримує відмову, зазвичай відмова виглядає як помилка доступу, але якщо системному адміністратору потрібно, він має можливість вносити налаштування та створювати повідомлення, яке буде приходити користувачу та повідомляти про те, чому саме він не може отримати доступ до ресурсу. Якоїсь великої цінності це повідомлення не має, тому зазвичай ним нехтують, або додають в випадках, коли користувачі масово скаржаться на проблеми з мережею, через які в них не завантажуються сайти. Також інструментом захисту користувачів, є організаційні заходи, наприклад, семінари чи лекції, на яких фахівці простою мовою пояснюють основи безпечного перебування в мережі та користування нею.

Захист інформації при її передачі за межі інформаційної системи підприємства є практично найбільш важливий та цікавий напрям захисту інформації. Його забезпечення відбувається за використанням інструментів одного або декількох з трьох варіантів:

- криптографія;
- стеганографія;
- захищені канали.

Організація захищених каналів реалізує створення своєрідного тунелю, по якому інформація передається від відправника до отримувача. Цей спосіб дозволяє відмовитись від передачі інформації через незахищений простір, але потребує часу та навичок від спеціаліста, який буде створювати цей канал та слідкувати за його роботою. Даний канал хоч і дає підвищений рівень безпеки при передачі даних, але все одно є шанс, що зловмисник зможе отримати до нього доступ, тому спеціаліст повинен не тільки створити та налаштувати зв'язок, а також і слідкувати за каналом після його створення.

Криптографія та стеганографія, по суті схожі між собою, це є способи, які беруть інформацію, яку необхідно передати, та змінюють її таким чином, щоб навіть при отриманні доступу до неї, користувач не міг її використати. Розглянемо відмінність між ними. Криптографічні інструменти – це, по суті, шифрування, тобто інформація після шифрування стає абсолютно нечитабельною для особи, яка не має ключів для її дешифрування. Тобто, якщо зловмисник отримає інформацію, яка була зашифрована криптографічним методом, то він отримає нечитабельний файл, він буде знати, що в ньому інформація, але не буде мати можливості її прочитати. У випадку зі стеганографією, вона також шифрує інформацію, але робить це таким чином, що зловмисник який отримає файл зашифрований стеганографічно, навіть не буде впевнений що це саме те, що він хотів отримати. Стеганографія приховує сам факт того, що існує якась інформація, зазвичай така інформація виглядає, як щось інше, наприклад зображення, або лист. Таким чином лише той, хто знає що тут є інформація, а також ключі для її дешифрування, зможе отримати доступ до того, що було сховано [19].

Таким чином представлені інструменти дозволяють захистити інформацію при обміні. Загалом вони зазвичай використовуються разом, оскільки при передачі інформації використовуючи стеганографію чи криптографію, також окрім інформації, потрібно передати отримувачу і ключ, завдяки якому він зможе цю інформацію розшифрувати. Але якщо цей ключ буде передаватись через незахищений простір, то є ймовірність перехоплення його зловмисником. Таким чином зловмисник може перехопити зашифроване повідомлення та ключ і буде мати все необхідне для дешифровки. Насправді ще є нюанси, пов’язані зі способами шифрування, та алгоритмами, які використовувались для нього, але в контексті даного питання ми їх упустимо. Тому щоб організувати захищену передачу даних, використовується шифрування та захищений канал зв’язку. Зашифроване повідомлення передається через незахищений простір, в той час як ключі – через захищений канал. Таке поєднання дозволяє максимізувати захист інформації [14].

Щодо інструментів, які використовуються для шифрування, вони бувають різних типів: як програмні, так і технічні. Наприклад, це може бути програма встановлена на ПК користувача, він вручну генерує ключ та шифрує повідомлення з його використанням, після чого здійснює відправлення, а може бути автоматизований прилад, який приймає трафік з конкретного ПК, та автоматизовано генерує ключ та організовує відправлення його та шифрованого повідомлення. Технічний засіб є більш дорого вартісним та складним в налаштуванні ніж програмний. Деякі з них взагалі є безкоштовними, але він дозволяє відділити користувачів системи від процесів шифрування, оскільки з одного боку, вони не мають необхідності цим займатись, а з іншого немає потреби навчати їх ще одній програмі, яка практично не пов’язана з їх робочою діяльністю.

2.3. Приклади загроз та протидій

В даному підрозділі, ми розглянемо приклади загроз, які можуть відбутись на типових підприємствах, а також те, яким чином, представлені вище інструменти дозволяють з ними впоратись, або ж мінімізувати шкоду причинену ними. Щоб урізноманітнити список загроз, будуть представлені три приклади, кожен з яких чинить загрозу на один з трьох основних принципів інформаційної безпеки:

- цілісність;
- конфіденційність;
- доступність.

Також варто розуміти, що оскільки всі принципи пов’язані між собою, то кожен приклад не чинить загрозу лише на один з них, відбувається порушення всіх трьох принципів, просто визначений принцип порушується максимально.

Загроза конфіденційності.

Як нам вже відомо, порушення конфіденційності інформації – це ситуація, коли доступ до інформації отримує особа, яка немає на нього права. В залежності від інформації, до якої був отриманий доступ, порушення конфіденційності може бути різного ступеню важливості. Наприклад, розглянемо ситуацію коли неавторизований користувач отримав доступ до листа, в якому керівник організації звертається до бухгалтерії та наводить список працівників, яким варто дати преміальну винагороду в цьому місяці, оскільки вони якісно виконували свою роботу. Така інформація конфіденційна, але не має великого значення. Вся користь, яку зловмисник може отримати з даного листа, це список сумлінних співробітників, тобто користь яку він отримає для себе є мінімальною. В іншому випадку зловмисник отримує доступ до логістичної системи організації та інформацію про її роботу з підрядниками, наприклад деталі угод, їх кошториси та контакти підрядників. В двох випадках це порушення конфіденційності, але якщо перший випадок не чинить для організації практично ніякої шкоди, другий у може принести критичні збитки, оскільки, якщо ця інформація потрапить до прямого конкурента, він може її використати для того щоб переманити підрядників, запропонувавши їм трохи кращі умови співпраці, а також створити перепони знаючи логістику своїх опонентів.

В якості реального прикладу загрози конфіденційності візьмемо простір, в якому система найбільш вразлива, – це за межами периметру, а також фактор, який неможливо повністю контролювати, це “людський фактор”. Тобто прикладом є крадіжка носія інформації за межами території підприємства. Нажаль такі ситуації відбуваються доволі часто, завдяки розвитку портативних технологій та технологій зв’язку. Працівники багато професій можуть виконувати свої обов’язки практично з будь-якої точки світу, а як показують останні події у світі, багато підприємств взагалі потроху відмовляються від фізичних офісів та максимально переводять робочий процес в інформаційний простір. Наявність такої тенденції в свою чергу провокує нові загрози, зокрема розглянуту тут загрозу конфіденційності. Розглянемо приклад коли працівник виконує свої робочі

обов’язки в себе дома, потім вирішує, що варто змінити обстановку, і йде в кав’ярню та бере з собою ноутбук. В кав’ярні він кудись відходить на декілька хвилин чи то до вбиральні, чи зробити замовлення, повертається та не знаходить свій ноутбук, оскільки його вкрали. Звісно першою справою він жалкує за самим пристроєм, але потім згадує, що на цьому пристрої також зберігається робоча інформація, а також з нього є відкритий доступ до ресурсів організації, її баз даних і т. п. По суті система авторизується не по людині, а по пристрою, з якого людина отримує до неї доступ. Таким чином для системи крадій і власник ноутбука сприймаються як одна особистість. Тепер розглянемо які засоби з представлених вище допоможуть вирішити цю проблему. По-перше, це організаційні заходи, на яких працівника навчили:

- для безпеки на пристрої повинна бути встановлена система авторизації, це може бути банальний пароль;
- коли користувач завершує роботу з пристроєм, він повинен перевести його в стан сну, найбільш поширені операційні системи працюють таким чином, що коли пристрій переведений в стан сну, то після його виведення, система запросить повторну авторизацію користувача;
- існують технічні засоби, які дозволяють відслідкувати свій пристрій, якщо він був попередньо підключений до іншого пристрою чи до спеціалізованого ресурсу.

Наведені вище заходи, самі по собі вже сильно ускладнюють доступ до інформації, яка зберігається на пристрої, а останній дозволяє простіше знайти злодія при зверненні до спеціалізованих органів.

По-друге, важливу роль виконують налаштування самої системи організації, наприклад, доступ до особливо важливих ресурсів, повинен періодично робити запит матеріалу для авторизації, наприклад ключа, чи пароля, та дозволяти подальшу роботу лише після його введення.

Щодо засобів авторизації, конкретно на пристрої користувача, то також існують програмні засоби, які шифрують вміст диску, в випадку, якщо декілька

разів до ряду був введений невірний пароль, ці засоби захищають від банального підбору пароля.

Загроза цілісності.

Прикладами загроз цілісності є несанкціоновані зміни внесені в інформацію, які призводять до проблем. Є можливим навести приклад з реального життя, в 1996 році працівниця однієї з компаній, яка замала пост секретаря віце президента компанії, подала судовий позов на нього, та звинуватила віце президента в незаконному звільненні через особисті причини, тобто, суть її позову була в тому, що вона прекрасно виконувала свої робочі обов'язки, а звільнена була через особисту неприязнь свого начальника, яка ніяким чином не відносилась до робочого процесу. Основним доказом з її боку, був лист, цього самого начальника до президента компанії, в якому якраз і була описана причина звільнення, в контексті, який доводив позицію секретаря. На листі, який був представлений в якості доказу, була позначення дата та час відправки, тоді в свій захист, віце-президент запросив у компанії стільникового зв'язку виписку по його телефону на вказаний день. В ній було вказано, що у вказаний час віце президент проводив телефонну розмову та знаходився за межами свого робочого місця. Таким чином ситуація переросла в протистояння двох файлів інформації: листа секретарші та виписки віце президента. Ситуація була вирішена доволі просто, коли піднялось питання того, звідки секретарша отримала доступ до листа, який був надісланий нібито віце президентом компанії до президента. Виявилось, що через свої робочі обов'язки, секретарша мала доступ до комп'ютера свого начальника, оскільки під час його відсутності замінювала його. Після цієї інформації суд відхилив позов, оскільки стало зрозуміло, що інформація в листі є несправжньою, а секретарша мала можливість її сфабрикувати, що доводила виписка зі компанії стільникового зв'язку.

Даний інцидент відбувся в 1996 році, коли технології були розвинуті набагато слабкіше ніж зараз. Давайте подивимось, як би події розвертались в 2022 році. По-перше, завдяки поширенню мобільних засобів, віце президент відразу б

дізнався про фабрикований лист, оскільки доступ до пошти він може отримати в будь-який момент, і може помітити нове відправлення, якого він не робив. По-друге, секретарша мала пароль від комп'ютера, і змогла отримати доступ до пошти, оскільки та була не захищена, а на даний момент практично всі поштові додатки потребують авторизації з особистого профіля користувача. Також, зараз, завдяки розповсюдженню та низькій вартості комп'ютерів, а також, тому, що на великих підприємствах усвідомлюють важливість безпеки, ситуація коли секретар знає пароль начальника є малоймовірною. Також є засоби відеоспостереження, які встановлюють в тому числі для контролю таких ситуацій.

Загроза доступності.

Зазвичай ці загрози призводять до технічної або фізичної неможливості системи виконувати свої функції. Прикладами таких загроз може бути атака на систему, яка перевантажує її можливості, через що вона виходить з ладу, а також поломка обладнання, нездатність, або небажання працівників виконувати свої функції і т. д. Загалом основна проблема цих загроз, це те, що організація не може продовжувати роботу в звичному режимі, та втрачає час, а час як відомо, є одним з найбільш цінних ресурсів, особливо для великих організації. Варто згадати лише 4 млрд. доларів збитків, які отримала компанія Meta, коли їх продукт Facebook став недоступним через помилку на декілька годин. Окрім фінансових втрат загрози доступності б'ють по репутації компанії, оскільки, якщо в компанії постійні проблеми з серверами, чи недоступністю їх ресурсів, то мало хто буде мати бажання працювати з такою компанією, яка в останній момент, може перенести встановлені терміни через технічні неполадки.

Розглянемо класичний варіант загрози доступності та варіанти його вирішення. Класична проблема доступності – це вихід системи з ладу, через вихід з ладу обладнання. Це є доволі поширена проблема, з якої рано чи пізно зустрічаються всі організації, незалежно від того, наскільки дорогу за вартістю техніку вони використовують. Техніка може підвести, наприклад вийти з ладу через стрибок напруги, або фізичне пошкодження. В такій ситуації технічний

спеціаліст, або адміністратор повинен мати алгоритм дій та резервне обладнання для швидкого вирішення проблеми. Ідеальним є варіант, коли резервне обладнання є підключеним у пасивному режимі, та входить до роботи в випадку поломки основного обладнання. У такому випадку адміністратор отримує сигнал про поломку та вирішує проблему, після чого система сама переключається на основне обладнання, повертаючи резервне в стан сну. Також важливим пунктом є забезпечення фізичної безпеки обладнання, тобто, встановлення перегородок та визначення безпечних зон для встановлення там мережевого обладнання, щоб наприклад випадковий працівник не міг під час перерви випадково пролити каву на сервер. Звісно, якщо у працівника буде стояти мета пролити каву на сервер, то сервер розташування не рятує. Тому у цьому випадку у дію вступають інші засоби фізичної безпеки, такі як замки та перегородки. В ідеалі, доступ до основного обладнання повинні мати лише обмежений перелік фахівців, які безпосередньо з ним працюють.

2.4. Проміжний висновок

При аналізі питань фізичної безпеки захисту інформації було розглянуто основні напрямки забезпечення безпеки на підприємстві, а також методології, які використовуються для побудови захисту. Зазвичай ці методології не впливають напряму на побудову захисту, вони допомагають провести тестування системи, або обрати необхідний варіант реалізації обмеження доступу, а сама комплексна система захисту інформації є настільки унікальною для кожного підприємства, що по суті не існує загальної методики для її побудови. Єдине, що об’єднує всі системи захисту інформації, це принципи захисту інформації, відповідно до яких і відбувається проектування, а в подальшому побудова системи. Також в даному розділі були розглянуті деякі приклади загроз, для кожного з принципів та представлено засоби, які дозволяють зменшити втрати у випадку їх реалізації.

3. МОДЕЛЮВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

В ході даного розділу буде розроблена модель системи, яка в подальшому буде реалізовуватись в наступному розділі. Цей розділ включає в себе IDEF, UML діаграми, моделі бази даних, а також розробку дизайну майбутнього додатку.

3.1. Моделювання системи

Першим кроком в моделюванні додатку є визначення функціональних вимог до нього, тобто необхідно створити перелік функцій, які повинні бути реалізовані в додатку. Додаток представляє з себе інструмент для візуалізації зв’язків між об’єктами та суб’єктами інформаційної системи, тобто є можливим сказати, що цей додаток, це свого роду панель адміністратора для спеціаліста, який займається інформаційною безпекою на підприємстві, він додає на цю панель елементи системи, виходячи з наявної в нього інформації, встановлює зв’язки між ними та отримує на виході структуру підприємства. Також варто звернути увагу на механізм допуску до додатку, оскільки додаток зберігає інформацію про працівників підприємства, то вільний доступ до нього є неприйнятним, таким чином необхідно реалізувати механізм авторизації користувача. Отже функціональні вимоги до додатку виглядають таким чином.

- Авторизація користувача
- Перегляд списку працівників та списку девайсів
- Можливість додавати нові записи до списку працівників та списку девайсів
- Можливість встановлювати зв'язок між працівником та девайсом

- Збереження всіх записів в БД
- Можливість переглянути які девайси є прив’язані до конкретного користувача та змінювати прив’язки за необхідності
- Редагування та видалення записів з бази через додаток.

Побудуємо діаграму IDF0(рис. 3.1.1.), яка узагальнює вимоги, та представляє візуальну картину додатку.

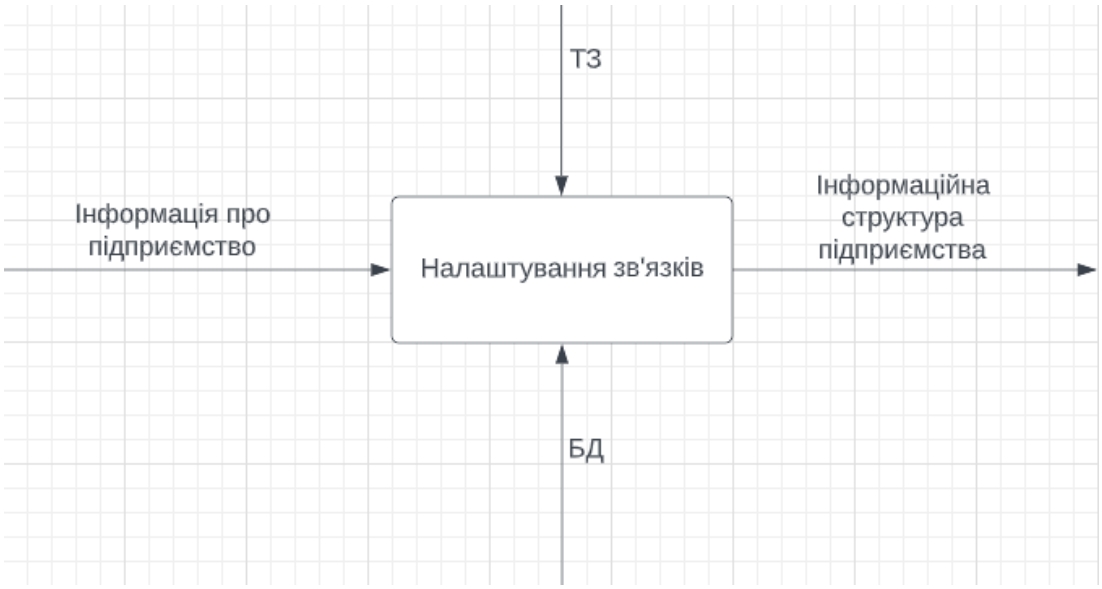


Рисунок 3.1.1. – Діаграма IDF0 системи підтримки інформаційної безпеки

Таким чином, виходячи з діаграми, ми бачимо загальну схему роботи додатку, а саме, користувач вносить інформацію про підприємство, а за допомогою таблиць БД та зв’язків між ними, формується вигляд інформаційної структури підприємства, завдяки якій користувач має можливість приймати рішення щодо безпеки підприємства та засобів, які варто організувати.

3.2. Моделювання варіантів використання

В контексті даного проекту, робота з додатком не передбачає неавторизованих користувачів, таким чином, для діаграми варіантів використання (рис. 3.2.1.) буде представлений лише один актор, який представляється авторизованим користувачем.

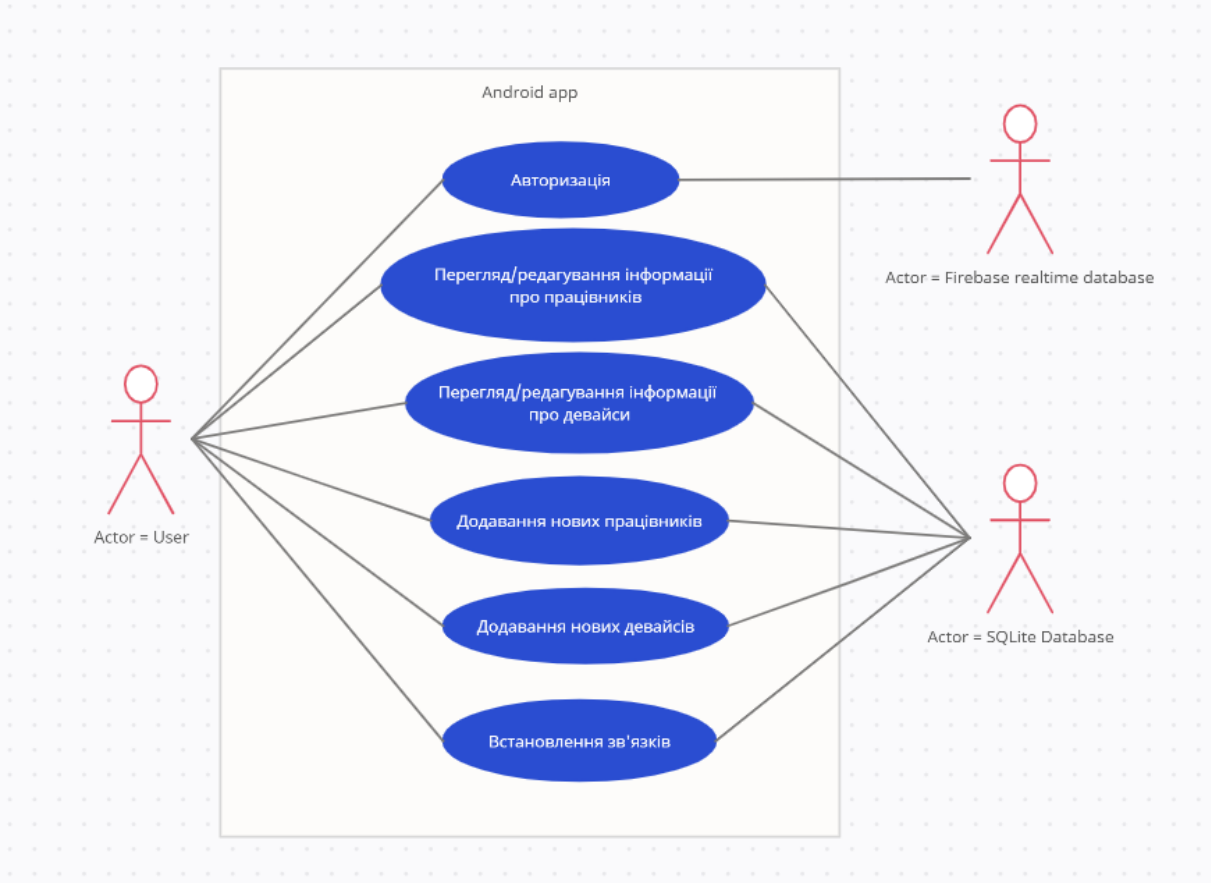


Рисунок 3.2.1. – Діаграма варіантів використання

Таким чином, на даний момент ми змоделювали внутрішні процеси додатку, а також варіанти його використання, одним з ключових моментів даного додатку є БД, яка зберігає всередині себе дані, а також дає користувачеві додавати в неї нові дані через інтерфейс додатку.

3.3. Проектування моделі бази даних

В контексті даного додатку, база даних складається з 4 таблиць:

- Workers – таблиця, яка зберігає інформацію про працівників організації (суб’єкти інформаційної безпеки)

- Devices – таблиця, яка зберігає інформацію про прилади для роботи з інформацією (об’єкти інформаційної безпеки)

- Link – таблиця, яка зберігає інформацію про встановлені зв’язки між таблицями Workers та Devices

- Users – таблиця, яка містить дані для авторизації.

Побудуємо ER діаграму (рис. 3.3.1.) для представлення зв’язків між таблицями наведеними вище.

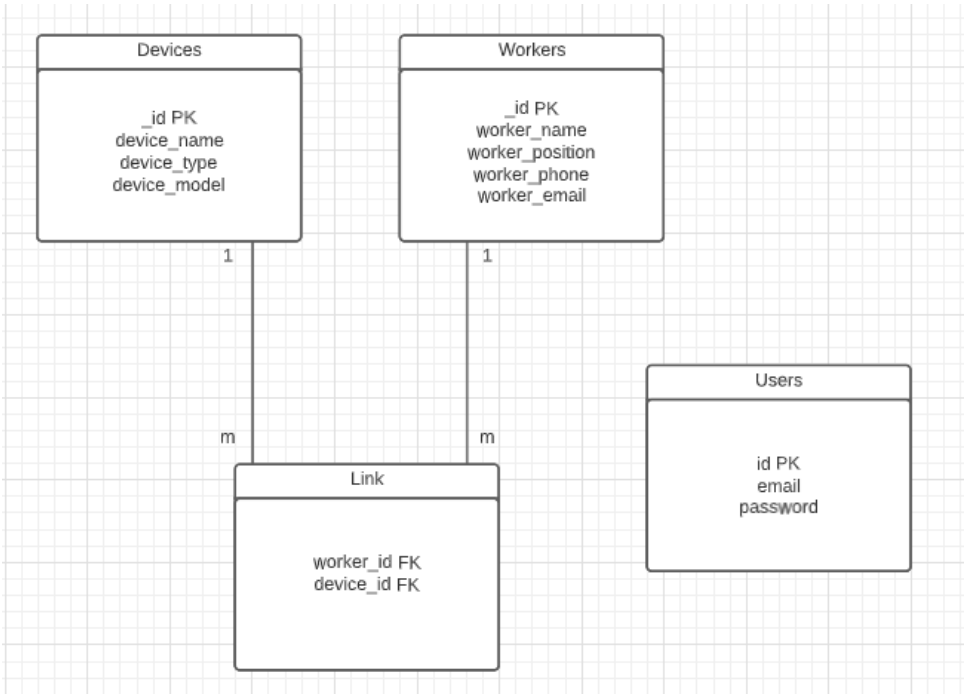


Рисунок 3.3.1. – ER діаграма

Розберемо детально кожну таблицю та її атрибути.

Таблиця Workers містить 5 полів, та представляє окремих працівників з їх персональними даними(таблиця 3.3.1.).

Таблиця 3.3.1. – Опис полів таблиці Workers

Назва	Тип	Опис
_id	Int	Ідентифікатор працівника, первинний ключ таблиці Workers
Worker_name	Text	Ім’я працівника
Worker_position	Text	Посада працівника
Worker_phone	Text	Телефон працівника
Worker_email	Text	Пошта працівника

Таблиця Devices містить 4 поля, та представляє девайси з їх основними характеристиками(таблиця 3.3.2.).

Таблиця 3.3.2. – Опис полів таблиці Devices

Назва	Тип	Опис
_id	Int	Ідентифікатор девайса, первинний ключ в таблиці Devices.
Device_name	Text	Назва девайсу
Device_type	Text	Тип девайсу(портативний чи стаціонарний)

Device_model	Text	Модель девайсу

Таблиця Link містить два поля, які представлені зовнішніми ключами, вона виступає для зв’язку попередніх двох таблиць, та дає можливість поєднувати дані з них за ключами(таблиця 3.3.3.).

Таблиця 3.3.3. – Опис полів таблиці Link

Назва	Тип	Опис
Worker_id	Int	Ідентифікатор задачі, первинний ключ таблиці Tasks.
Device_id	Int	Ідентифікатор девайсу, зовнішній ключ, через який таблиця Devices пов’язується з таблицею Tasks.

Та останнім йде огляд таблиці Users, як видно на рисунку(рис. 3.3.1.), ця таблиця не є пов’язаною з рештою, вона використовуються для авторизації конкретного користувача, і надає доступ до функціоналу додатку, також її реалізація відрізняється від решти, оскільки попередні таблиці реалізуються з використанням вбудованих в середовище розробки SQLite інструментів, а також окремого SQLite Browser, який використовується для створення запитів на створення та тестування таблиць. Авторизація в свою чергу реалізується за допомогою Firebase, тобто дані з попередніх таблиць зберігаються напряму на девайсі з додатком, а таблиця Users для авторизації в хмарному сховищі.

Таблиця 3.3.4. – Опис полів таблиці Users

Назва	Тип	Опис

id	Int	Ідентифікатор запису для авторизації, первинний ключ таблиці Users.
email	Text	Пошта користувача

Таблиця 3.3.4. – Опис полів таблиці Users

password	Text	Пароль

Тепер ми маємо описаний функціонал, а також схему БД, яка необхідна для розробки додатку, таким чином виходячи з даних отриманих в цьому розділі є можливим приступити до розробки, залишається лише останній крок, це створення дизайну для додатку, маючи на руках його разом з структурою БД та функціоналом, залишиться лише вирішити технічні питання розробки.

3.4. Розробка дизайну

Першим кроком створимо схему, на яку нанесемо всі необхідні екрани додатку, які потребують розробки дизайну для них. Елементами цієї схеми стануть:

- Splash screen
- Екран авторизації
- Головний екран додатку
- Екран додавання працівника
- Екран девайсів
- Екран додавання девайсу
- Екран додавання девайса для працівника

- Екран редагування запису працівника
- Екран додавання запису девайса
- Екран перегляду інформації про працівника та його девайси
- Фрагмент для виводу статистики

В результаті ми маємо 11 екранів, для кожного з яких необхідно створити дизайн, а також розібрати взаємодію між цими екранами.

Для створення дизайну був використаний онлайн інструмент proto.io, який дозволяє розробляти прототипи для додатків різної направленості, та окрім створення дизайну, також дає можливість встановити зв’язки між екранами, та подивитись, яким чином вони будуть взаємодіяти між собою[22][23].

Першим екраном, який користувач бачить при запуску додатку є SplashScreen(рис. 3.4.1.), на ньому наводиться назва додатку, логотип, фраза з описом та шкала завантаження, в контексті даного додатку ця шкала є лише візуальним ефектом, оскільки додаток не потребує часу на завантаження додаткових даних, чи зв’язку з мережевими ресурсами.

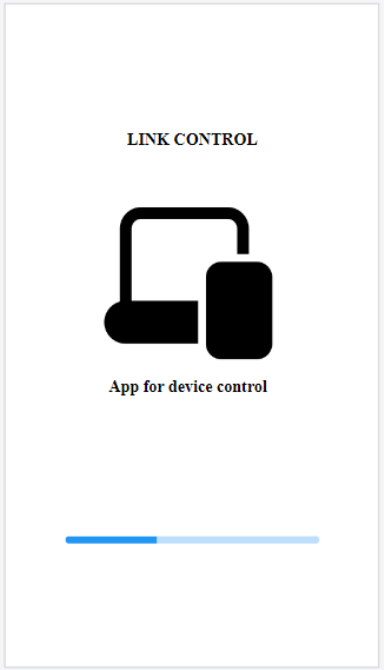


Рисунок 3.4.1. – Дизайн екрану SplashScreen

Наступним екраном, який бачить користувач э екран авторизації(рис. 3.4.2.), на даному екрані він повинен ввести свої дані для авторизації, щоб отримати доступ до основного функціоналу додатку.

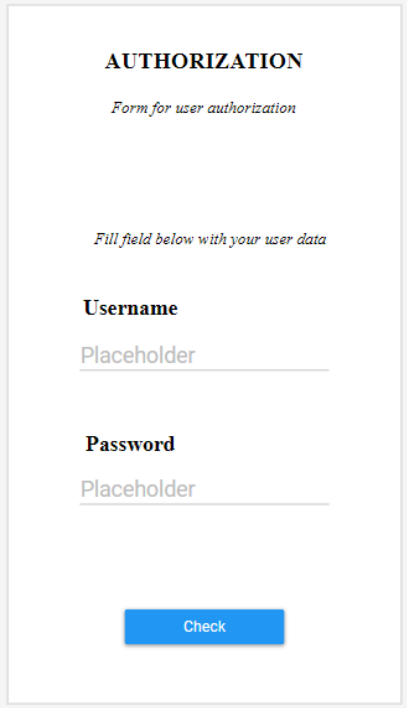


Рисунок 3.4.2. – Дизайн екрану авторизації

Після успішної авторизації в додатку, користувач потрапляє на основний екран(рис. 3.4.3.), на цьому екрані наведений список всіх працівників та їх даних, які внесені в БД, також, окрім цього списку, користувач бачить фрагмент статистики в нижній частині екрану, даний фрагмент показує статистичні дані на даний момент, а саме, кількість працівників, кількість девайсів, а також кількість девайсів яким були назначені працівники. Також на цьому екрані є 4 інтерактивні елементи, це кнопки для редагування чи видалення інформації про працівника, ці кнопки прив’язані окремо до кожного запису, елемент для переходу на активність додавання нового працівника, у вигляді круглої кнопки в нижньому куті та елемент в для переходу до активності девайсів, у верхньому куті, на рисунку дизайна він

представлений іконкою макбука, в фінальній версії продукту дизайн іконки буде змінений.

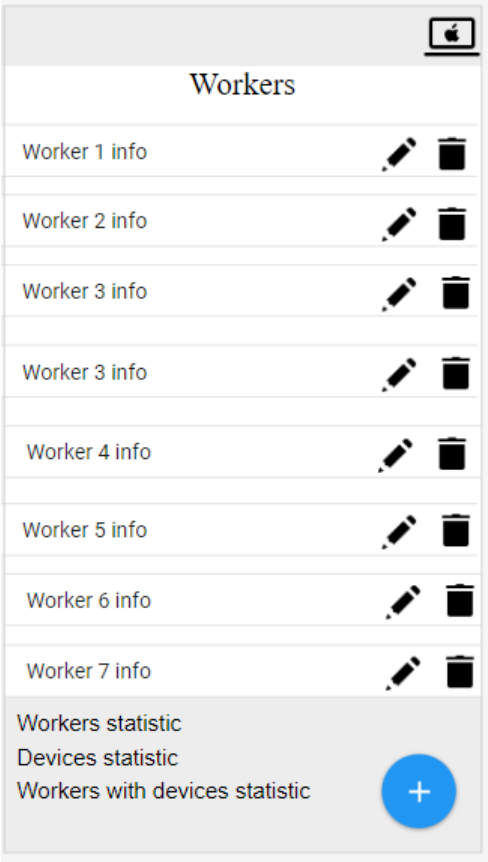


Рисунок 3.4.3. – Дизайн головного екрану

При необхідності внесення змін в існуючий запис, користувач натискає на іконку олівця, і перед ним відкривається вікно внесення змін(рис. 3.4.4.), в якому знаходяться поточні дані обраного запису, користувач вносить зміни та при натисканні на кнопку Upload зміни зберігаються в базу.

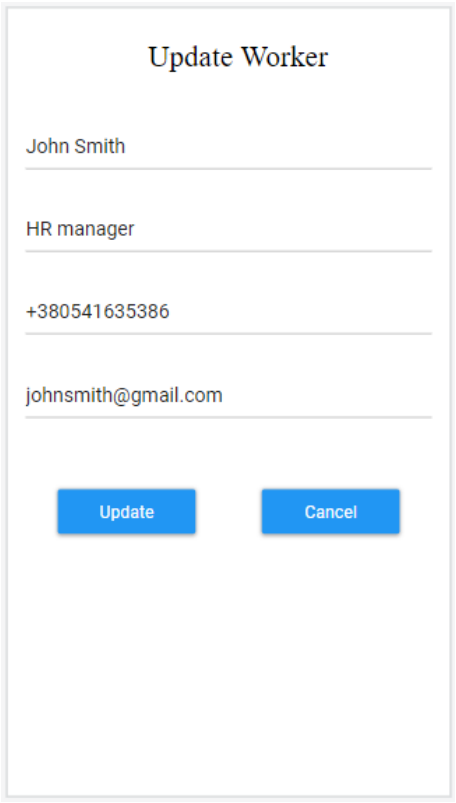


Рисунок 3.4.4. – Дизайн вікна редагування працівника

Якщо ж користувачу необхідно додати нового працівника до бази, він натискає на кнопку в нижньому куті та потрапляє до вікна створення нового працівника(рис. 3.4.5.). В цьому вікні користувач бачить поля, які потрібно заповнити, а також заголовки, які показують, які саме дані потрібно ввести в конкретне поле.

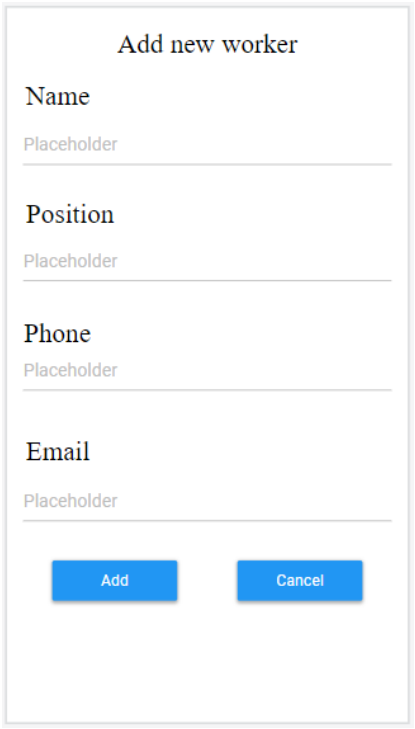


Рисунок 3.4.5. – Дизайн вікна додавання працівника

Активність видалення запису з бази не потребує освітлення, оскільки користувач натискає на відповідний елемент, додаток відкриває діалогове вікно з підтвердженням видалення, і якщо користувач підтверджує видалення, елемент видаляється.

При натисканні на елемент в верхньому куті, користувач переходить з головного екрану, на екран девайсів(рис 3.4.6.), на цьому екрані наведений список існуючих девайсів, з кнопками редагування та видалення, статистичним фрагментом в нижній частині екрану, а також кнопкою додавання нового девайсу в нижньому правому куті, окрім цього в верхньому лівому куті наведена стрілка, яка дозволяє повернутись до головного екрану.

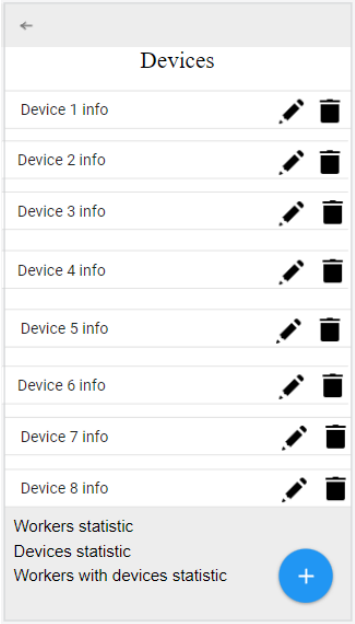


Рисунок 3.4.6. – Дизайн екрану девайсів

При потребі, користувач натискає на круглу кнопку в нижньому правому куті, та потрапляє до вікна створення нового девайсу(рис. 3.4.7.), в цьому вікні наведені поля, які користувач має заповнити, та дві кнопки, одна з яких додає новий девайс до бази, а інша відміняє операцію створення та повертає користувача на екран девайсів.

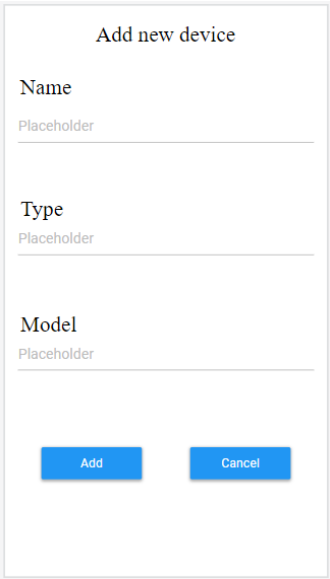


Рисунок 3.4.7. – Дизайн вікна додавання девайсу

Якщо користувачу необхідно відредагувати існуючий девайс, то він натискає на олівець, та потрапляє до вікна редагування інформації про девайс, функціонал такий самий, як з релагуванням даних працівника, користувач отримує вікно з введеними існуючими даними, вносить зміни та після натискання на кнопку ці зміни приймаються базою.

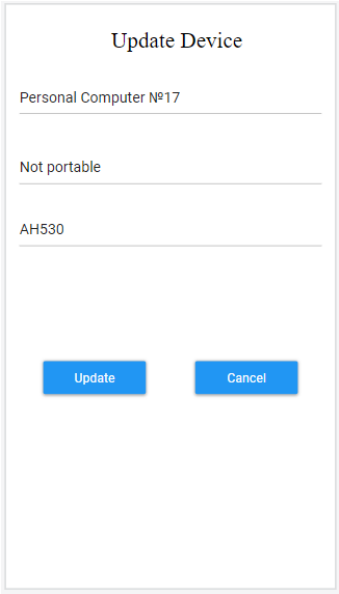


Рисунок 3.4.8. – Дизайн вікна редагування девайсу

Тепер, коли були розглянутий дизайн базових можливостей, є можливим перейти до дизайну екранів, які дозволяють пов’язувати працівника з девайсом, для цього повертаємось до головного екрану, та натискаємо на одного з існуючих працівників, після натискання на працівника ми потрапляємо до екрану з інформацією про цього працівника(рис 3.4.9.), тут дублюється інформація, яка наведена на головному екрані, а також наводиться список всіх девайсів, які є прив’язані до цього користувача, окрім цього, даний екран має три інтерактивних елемента, а саме це стрілка на тулбарі, яка дозволяє повернутись на головний екран, кнопка видалення девайсу для конкретного працівника, варто зауважити, що при натисканні на неї видаляється не девайс, а зв'язок між ним та конкретним працівником, та кнопка додавання девайсів для працівника, яка знаходиться з

правого боку панелі з титулом області з девайсами, та в контексті дизайну виглядає як плюсик. А також в нижній частині розташований фрагмент зі статистикою.

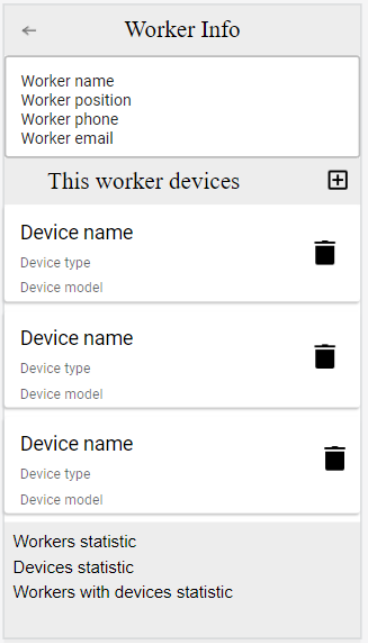


Рисунок 3.4.9. – Дизайн екрану інформації про працівника

При натисканні на плюсик, користувач має можливість додати девайси до працівника(рис 3.4.10.), чия інформація була відкрита.

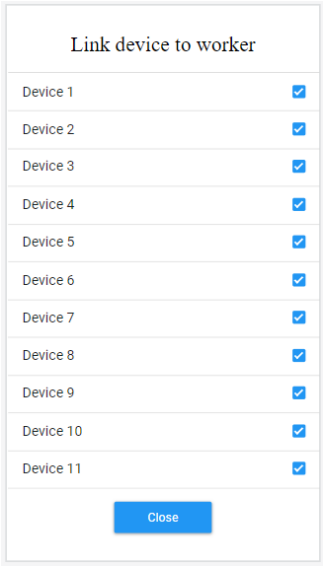


Рисунок 3.4.10. – Дизайн екрану додавання девайсів до користувача

Та фінальним екраном з дизайном є фрагмент статистики(рис 3.4.11.), який наводився як частина багатьох екранів раніше.

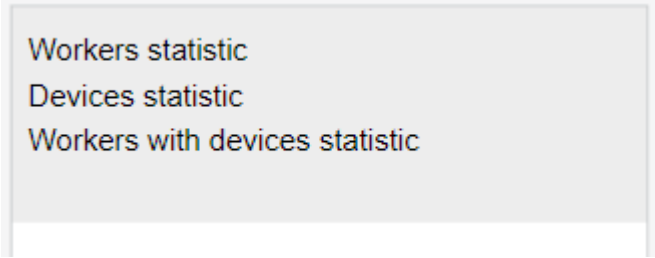


Рисунок 3.4.11. – Дизайн фрагменту статистики

3.5. Проміжний висновок

В ході створення даного розділу були розроблені діаграми, які демонструють процеси, які будуть виконуватись розробленим додатком, а також функціональні вимоги, які дозволяють будувати уявлення про функціональну частину додатку. Окрім цього, оскільки ключовою частиною проекту є база даних, було розроблено її концепт, а також схема її структури, з детальним описом таблиць і взаємодії між ними. Фінальним етапом даного розділу стала розробка дизайну додатку, яка дозволила візуалізувати процеси та більш чітко уявити екрани та зв’язки між ними.

В результаті, ми маємо опис процесів, схему БД, а також дизайн додатку, останнім кроком до його реалізації є лише вибір інструментарію для розробки, розробка технічної частини та тестування роботи додатку.

4. РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1. Інформаційна технологія

Комплексна система захисту інформації в рамках підприємства може набувати різних форм, та реалізовуватись з використанням широкого спектру різнопланових засобів, що є зрозумілим виходячи з матеріалів наведених в першому та другому розділах, але існують певні процеси, забезпечення яких є неймовірно важливою частиною інформаційної безпеки. По суті своїй ці процеси є ядром, навколо якого, та на основі результатів якого в подальшому додають нові модулі, нове програмне забезпечення, нові процеси, які покращують безпеку в напрямку необхідному конкретному підприємству.

В контексті даної роботи, було проведено виведення цих процесів та представлення їх у вигляді списку етапів, які є основними для подальшого проектування, та на основі яких і відбувається подальше вдосконалення системи, список цих етапів, в поєднанні з додатком, який покриває функціонально частину з них, є можливим назвати інформаційною технологією проектування комплексної системи захисту інформації на підприємстві.

Цей список етапів, виглядає таким чином:

- Етап 1 – Аналіз підприємства

  Даний етап проводиться в першу чергу, при виконанні практично будь-яких дій з інформаційною системою підприємства, починаючи від розробки маркетингової стратегії та закінчуючи побудовою системи захисту. На даному етапі проводиться аналіз бізнес-процесів, які відбуваються всередині підприємства, аналіз ієрархії, розділення обов’язків, дослідження діяльності компанії, тобто після виконання

даного етапу, отримується повна модель підприємства та його діяльності, що в подальшому використовується для прийняття рішень з приводу доцільності, чи недоцільності інтеграції засобів в контексті конкретно цього підприємства.

- Етап 2 – Визначення об’єктів інформаційної системи

  Як є відомо, будь-яка інформаційна система складається з об’єктів, суб’єктів та зв’язків між ними, а їх контроль є не те що доцільним, він є необхідним для забезпечення інформаційної безпеки на підприємстві. В ході цього етапу виділяються саме об’єкти, об’єктом є саме інформація, але робота з інформацією не відбувається напряму, люди для цього використовують посередників у вигляді девайсів які мають можливість, записувати, передавати чи відображати інформацію, тому в контексті інформаційної безпеки об’єктами виступають саме ці девайси. В результаті даного етапу, отримується список всіх об’єктів, які працюють в рамках інформаційної системи підприємства.
- Етап 3 – Визначення суб’єктів інформаційної системи

  Суб’єктами інформаційної системи є люди, які працюють з об’єктами, визначеними в ході попереднього етапу. По суті на даному етапі створюється список всіх працівників компанії, які мають доступ до інформаційної системи, та мають можливість чинити на неї вплив. Варто зауважити, що наприклад в контексті будівельної компанії, існують працівники, які не взаємодіють з системою напряму, це можуть бути будівельники, які отримують задачу від керівництва та займаються її виконанням, а дані до системи про ці процеси заносяться відповідальними за це працівниками.
- Етап 4 – Визначення зв’язків в інформаційній системі

  На попередніх двох етапах, було визначено списки об’єктів та суб’єктів системи, що вже само по собі дає певний простір для контролю над ними,

але без встановлених зв’язків, цей контроль не є повним, та залишається багато прогалин в роботі системи безпеки, наприклад за допомогою програмного забезпечення для моніторингу трафіку, є можливим помітити, що з конкретного девайсу було здійснено несанкціоновану передачу даних поза межі системи, таким чином спеціаліст відповідальний за безпеку знає що відбувся факт передачі, знає з якого девайсу це відбувалось, але не знає хто саме є прив’язаним до цього девайсу. Визначення зв’язків, в свою чергу пов’язує працівника та девайси з якими він працює, і робить ці девайси його зоною відповідальності, оскільки у випадку ситуації, коли з його девайса було здійснено дії, які можуть зашкодити підприємству, він буде відповідати за це. Таким чином з одного боку зв’язки дозволяють сформувати цільну структуру підприємства, а з іншого підштовхують працівників до дотримання правил інформаційної безпеки при використанні своїх девайсів.

- Етап 5 Проведення організаційних заходів для персоналу

  Після реалізації попереднього етапу, працівники самі стають зацікавленими в тому, щоб їх девайси були захищені належним чином від неавторизованих дій з ними, але в той самий час, їм необхідна інформація про те, як їх захистити, чи яких правил потрібно дотримуватись при роботі з ними. Для цього був визначений даний етап, який полягає в проведенні організаційних заходів для працівників, на цих заходах працівникам пояснюють елементарні засоби захисту в інформаційному просторі, яких правил варто дотримуватись, а чого робити не варто. Дані заходи не спрямовані на надання працівникам спеціалізованих знань, це лише елементарний курс для того, щоб працівник мав можливість на мінімальному рівні забезпечити безпеку своєї зони відповідальності та не створював небезпеку для всієї системи необачними діями.
- Етап 6 – Аналіз ризиків

Етап, який використовується коли базові етапи по забезпеченню захисту вже реалізовані, він полягає у визначенні можливих загрозу, прогнозуванню можливої шкоди та проведенню процесу оцінки. Таким чином отримується статистична інформація щодо ризиків та визначається найбільш небезпечні з них за відношенням шкода – ймовірність, після чого приймаються висновки щодо дій, які доцільно прийняти по відношенню до кожного ризику, спектр цих дій коливається від ігнорування до максимально можливої ліквідації ризику.

- Етап 7 – Інтеграція засобів захисту

  Після процесу оцінки ризиків та вибору з них найбільш небезпечних, приймаються відповідні дії, які зазвичай полягають в інтеграції певних засобів, програмних чи інженерних до системи, задля того, щоб знизити шанс виникнення ризику, або ж для того, щоб зменшити потенційну шкоду, завдану ризиком.

В контексті даної роботи здійснюється розробка мобільного додатку, який дозволяє покривати функціонал наведений в 2,3 та 4 етапах даної інформаційної технології [29],[32].

4.2. Вибір інструментарію

Базою додатку, була обрана система android, через свій високий рівень популярності, а також той факт, що існує комфортне середовище для розробки додатків на базі цієї операційної системи [25]. Цим середовищем розробки є Android Studio, його комфортність забезпечується зручним інтерфейсом, а також вбудованим емулятором, завдяки цьому є можливість писати код і відразу без затримок перевіряти, яким чином його робота буде виглядати на телефоні. Також однією з безумовних переваг даного середовища є його поширеність, завдяки їй

існує багато ресурсів в мережі Інтернет, на яких можна знайти корисну інформацію щодо розробки, а також безліч форумів на яких є можливим знайти готові відповідь на свої запитання, що в свою чергу дозволяє вирішувати практично будь-яку проблему, яка може виникнути при розробці. Нижче наведений інтерфейс Android Studio (рис. 4.2.1).

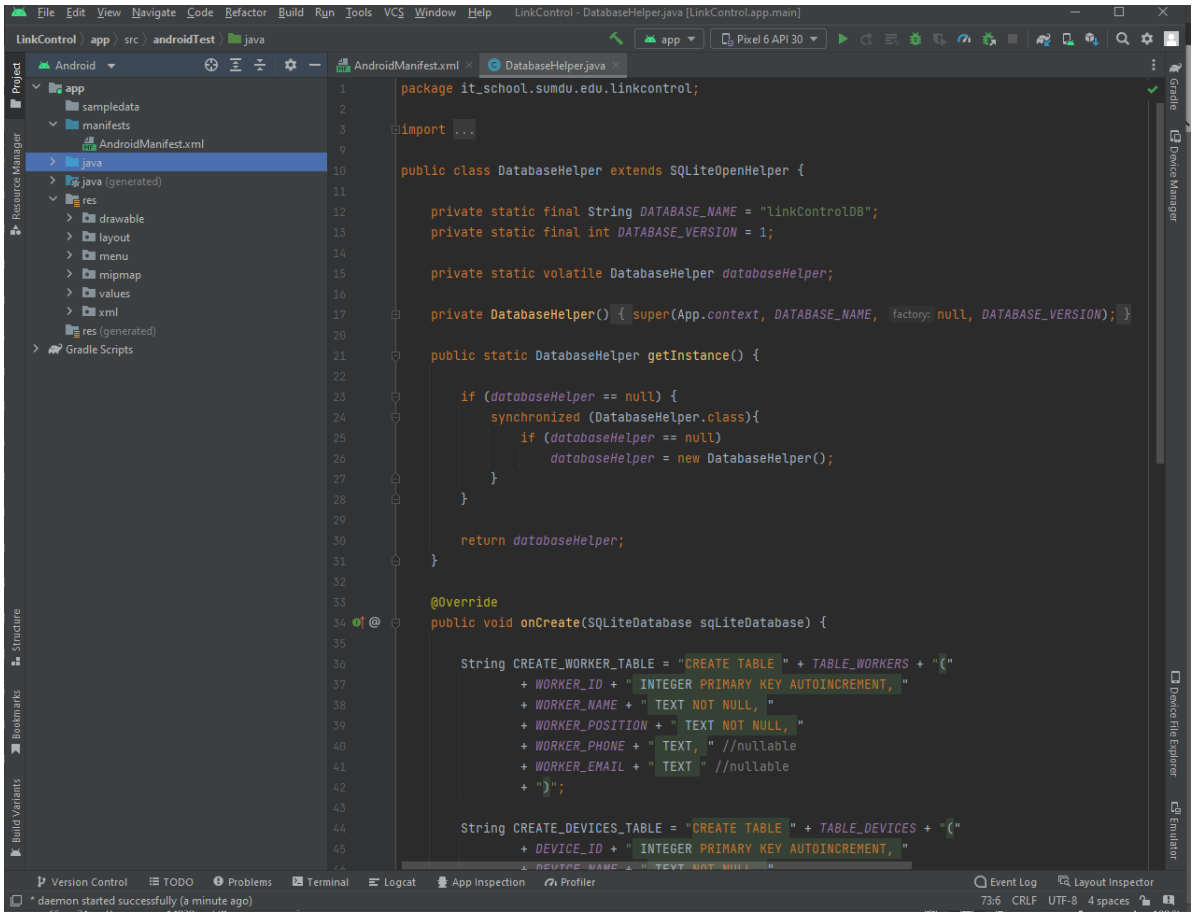


Рисунок 4.2.1 – Інтерфейс Android Studio

Важливим аспектом роботи в контексті розроблюваного додатку, є робота з базами даних, Android Studio має вбудований функціонал для роботи з SQLite[27], але при роботі з базами даних лише через Android Studio є доволі складно проводити тестування запитів та перевірку працездатності створених таблиць, тому для допомоги в цьому питанні, був використаний такий інструмент, як DB Browser SQLite(рис. 4.2.2), за допомогою цього інструменту було створено таблиці, відповідно до вимог поставлених в третьому розділі, після чого було проведено

тестування роботи запитів до цих таблиць на коректність, та в результаті, SQL запити для створення таблиць були імпортовані безпосередньо в Android Studio для того щоб виконувати з ними подальшу роботу, вже знаючи що вони працюють саме таким чином, як необхідно в контексті даного додатку.

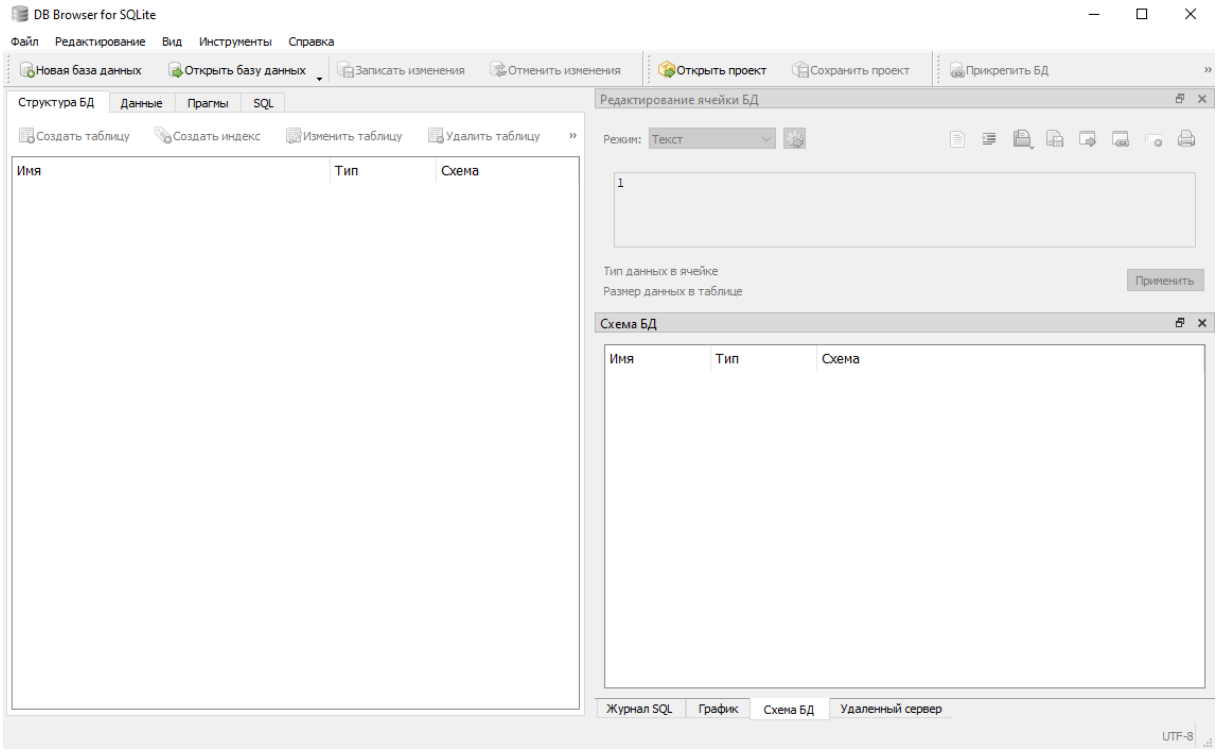


Рисунок 4.2.2 – Інтерфейс DB Browser SQLite

За допомогою представлених вище інструментів є можливо реалізувати весь функціонал системи, але при такій реалізації, функціонал авторизації буде дуже ненадійним, оскільки якщо реалізувати авторизацію через SQLite, то дані для неї будуть зберігатись напряму на девайсі користувача, що по суті зводить її користь на 0, оскільки отримавши доступ до телефону користувача, неавторизований користувач відразу має і базу даних для авторизації, через що отримати доступ до додатку не є важкою задачею. Тому для реалізації авторизації було обрано інструмент Firebase (рис. 4.2.3), цей інструмент дає можливість створити авторизацію користувача через NoSQL Realtime Database, дані для авторизації зберігаються в таблиці на Firebase, таким чином ризик отримання даних для

авторизації несанкціонованим користувачем є мінімальним. На рисунку нижче, наведений інтерфейс Firebase, в контексті одного з минулих проектів[31].

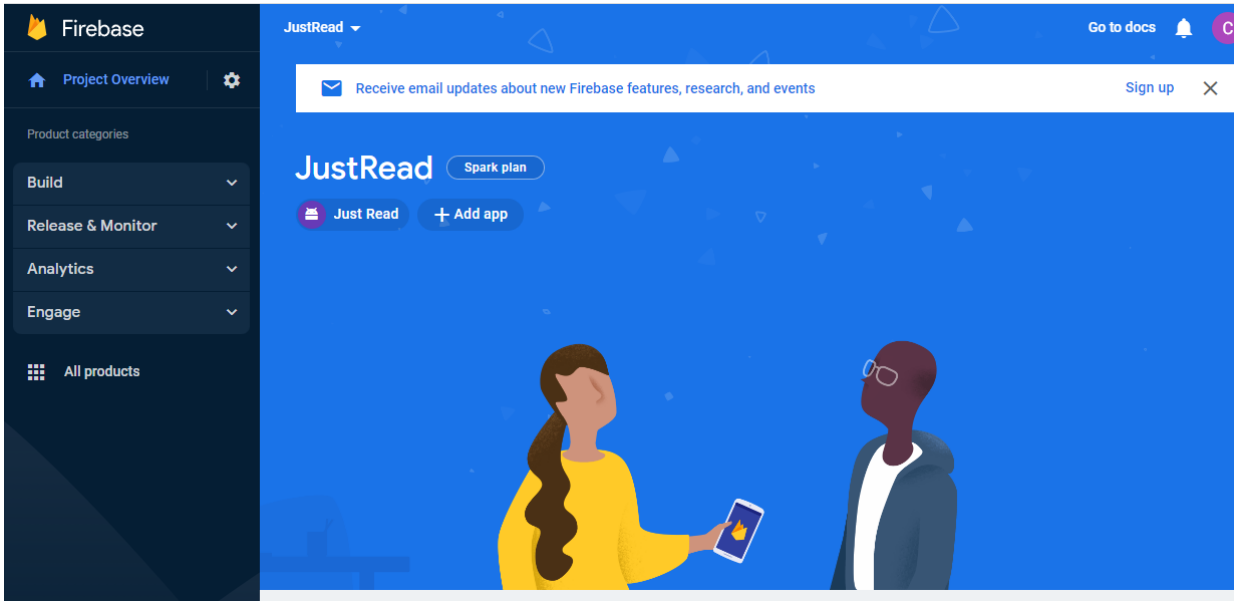


Рисунок 4.2.3 – Інтерфейс Firebase

Інструментарій для реалізації додатку обраний, тому є можливим перейти безпосередньо до розробки, а саме, реалізації БД, по моделі представленій в розділі номер три.

4.3. Реалізація БД

Як було сказано в попередньому розділі, в контексті даного проекту використовуються дві БД:

- SQLite для зберігання інформації та надання користувачеві можливості працювати з нею.
- Firebase для авторизації користувача

Для того, щоб реалізувати авторизацію користувача, для початку необхідно пов’язати проект, створений на Android Studio та Firebase, для цього в Firebase

створюємо новий проект(рис. 4.3.1.) та додаємо до нього додаток, після чого додаємо до нього авторизацію(рисунок 4.3.2.) , створений в середовищі Android Studio, на рисунках червоним контуром виділені основні моменти.

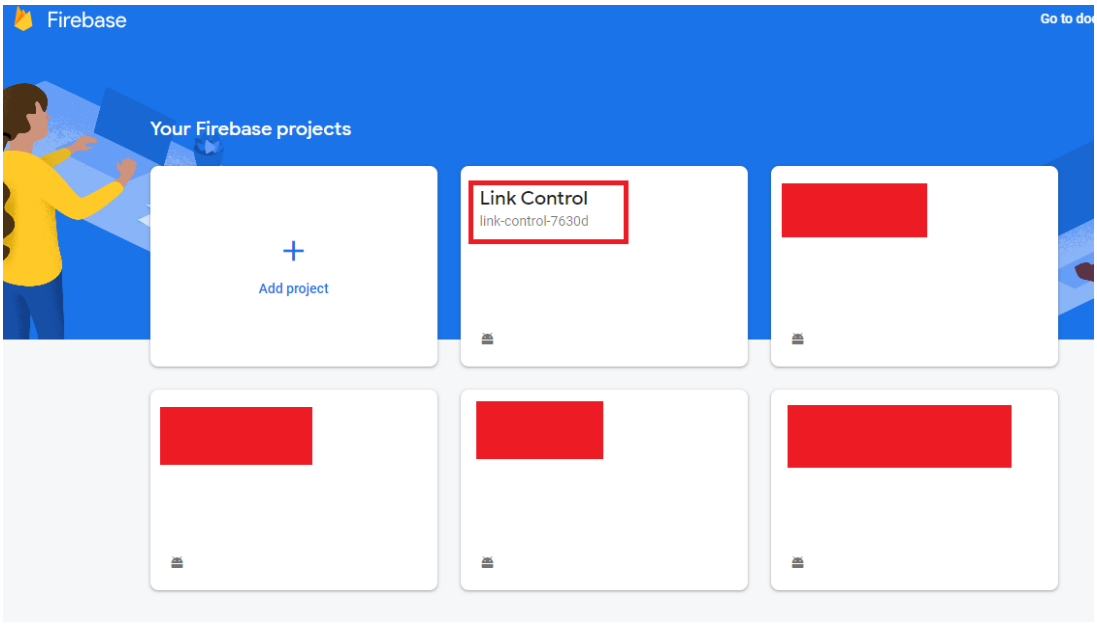


Рисунок 4.3.1 – Створення проекту в Firebase

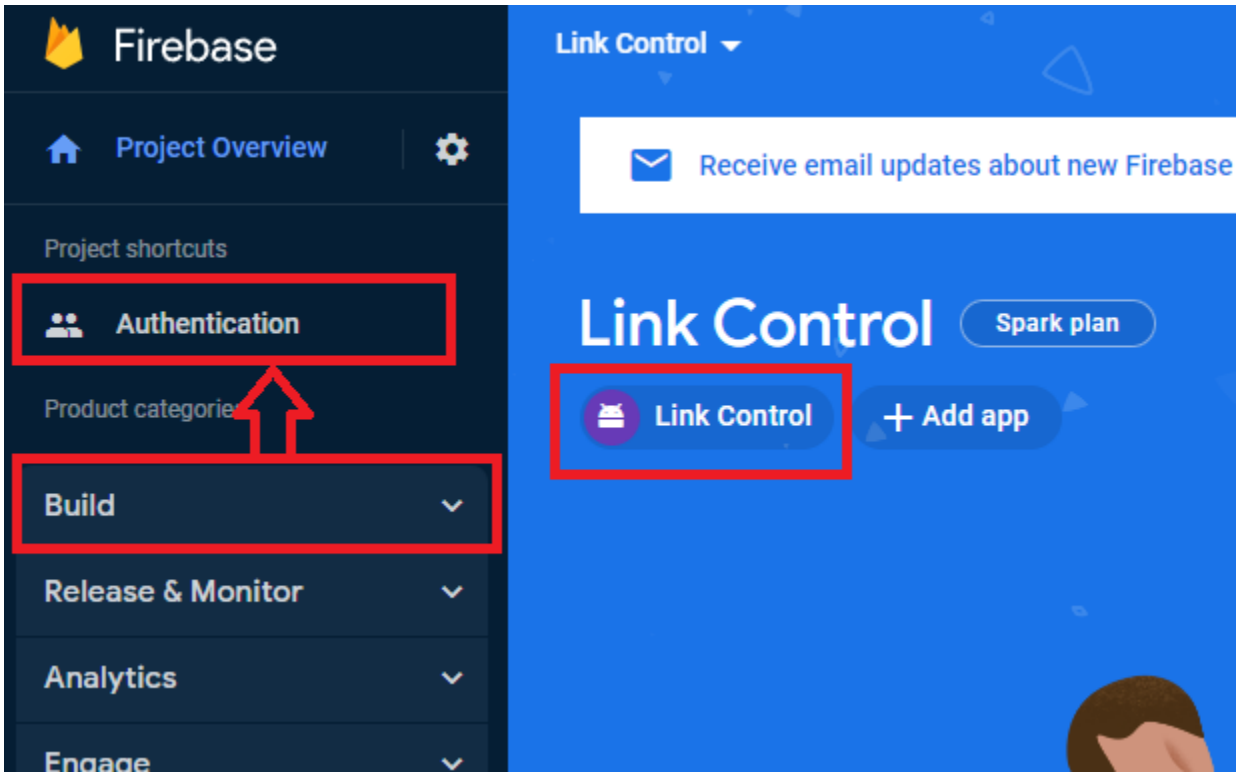


Рисунок 4.3.2 – Додавання додатку та авторизації

На вкладці Автентифікація(рисунок 4.3.3.), є наявна таблиця користувачів, з якою відбувається зв'язок, для авторизації чи реєстрації, в контексті даного проекту, відбувається лише перевірка ідентичності даних введених користувачем, з тими даними, які зберігаються в таблиці, та у випадку якщо ці дані є ідентичними, користувач допускається до наступної активності. Запис до цієї таблиці був доданий вручну, оскільки в контексті додатку процес реєстрації, який заповнює таблицю, не буде реалізованим.

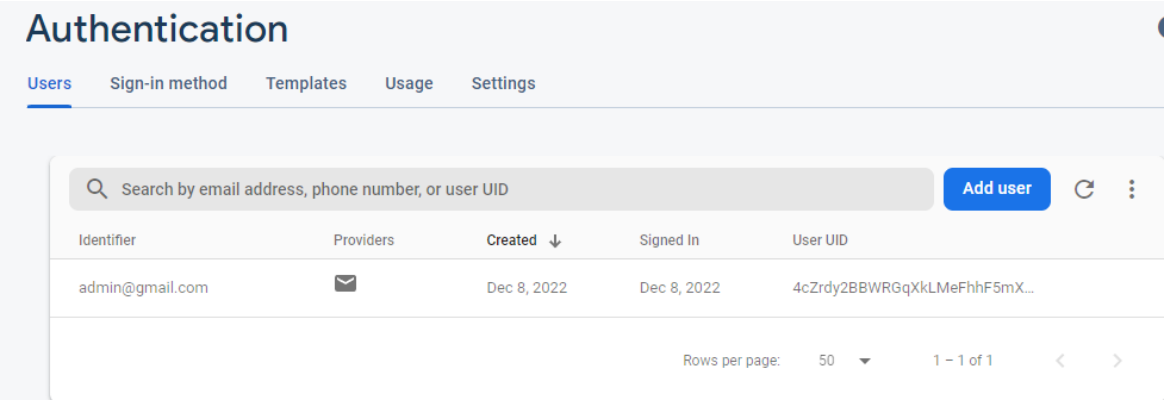


Рисунок 4.3.3. – Таблиця користувачів

На цьому етапі, робота з авторизацією зі сторони Firebase є закінченою, наступним кроком є додавання необхідних залежностей в Android Studio та синхронізація середовища з Firebase. Для цього, спочатку обираємо в інструментах Firebase та функціонал, який нам необхідний, в контексті даного проекту це автентифікація через користувальницьку систему(рис. 4.3.4.), після чого ми отримуємо список етапів, які потрібно виконати для реалізації даного функціоналу(рис. 4.3.5.).

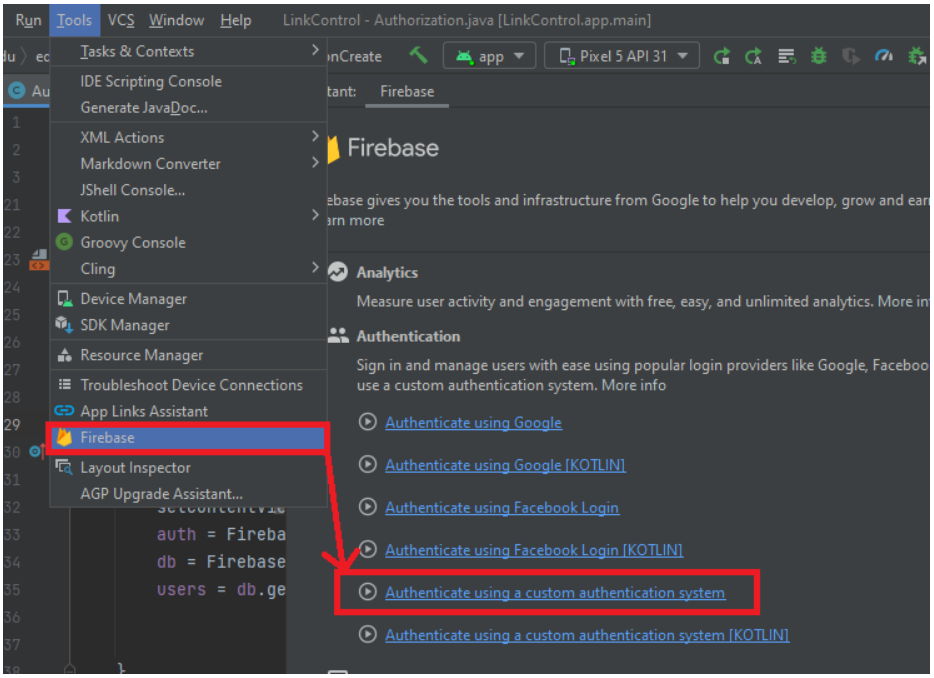


Рисунок 3.4.4 – Вибір функціоналу до реалізації

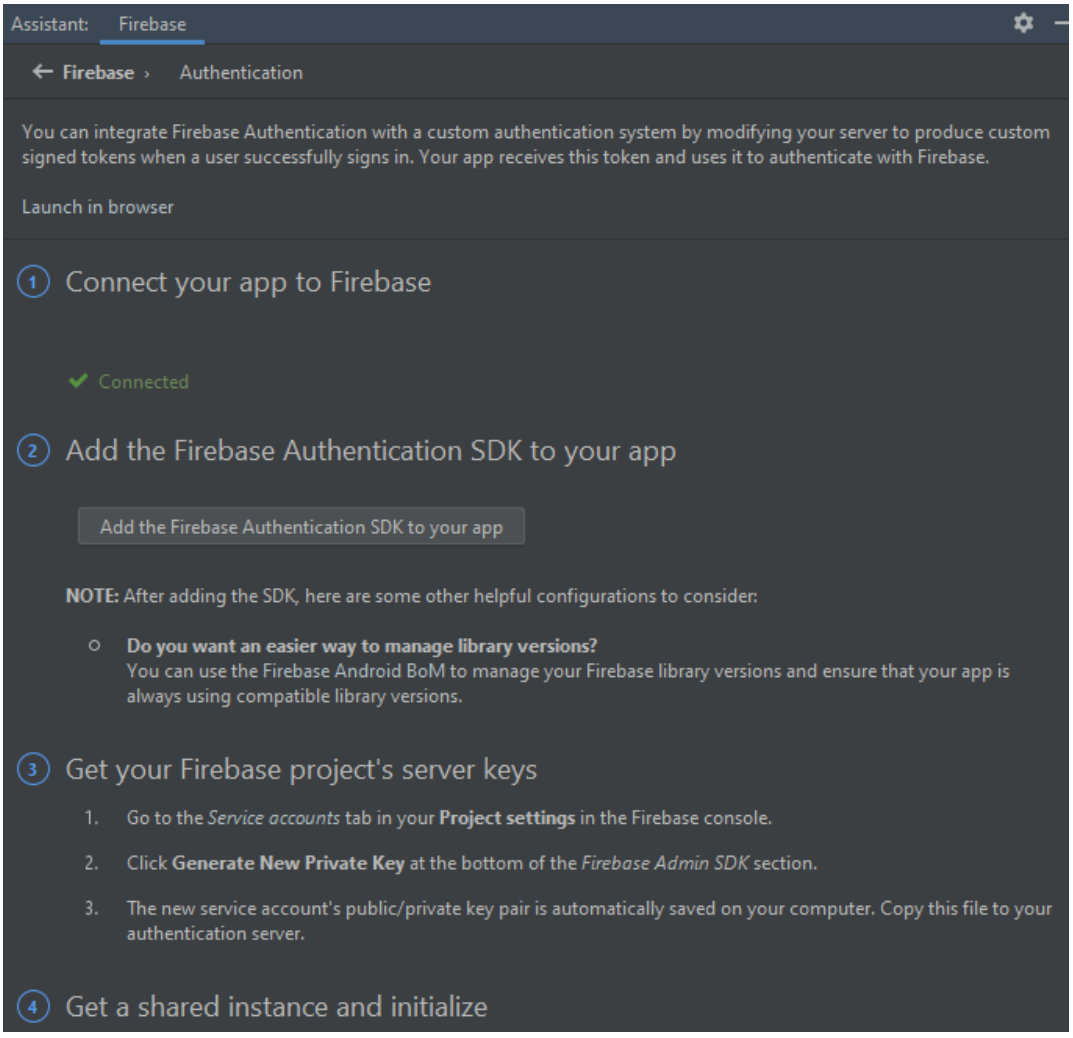


Рисунок 3.4.5 – Етапи реалізації

Після виконання всіх цих етапів, база є синхронізована з додатком та є можливим організовувати авторизацію користувача. Тобто робота з БД Firebase в контексті даного проекту є завершена, залишається лише реалізувати візуальний інтерфейс авторизації та викликати необхідні функції, які були наведені в етапах(рис. 3.4.5).

Далі йде організація роботи з SQLite DB, як було сказано в попередньому розділі, Android Studio підтримує SQLite та має вбудовані інструменти для роботи з ним, але робота за використанням лише цих інструментів, є доволі незручною, оскільки тестувати створення таблиць та запити до них через програмний код є доволі незручним, та може привести до помилок при реалізації навіть добре спроектованої бази. Таким чином для вирішення цього питання було обрано допоміжний інструмент, SQLite DB Browser, даний інструмент дає можливість створювати таблиці та тестувати їх за допомогою зручного інтерфейсу, а також в подальшому експортувати запити в Android Studio, таким чином за допомогою цього інструмента є можливим бути впевненим що запити експортовані в середовище розробки працюють саме таким чином, який є необхідний для розробника. Використовуючи схему, побудовану в розділі моделювання створимо відповідні таблиці в середовищі SQLite DB Browser(рис. 4.3.6.).

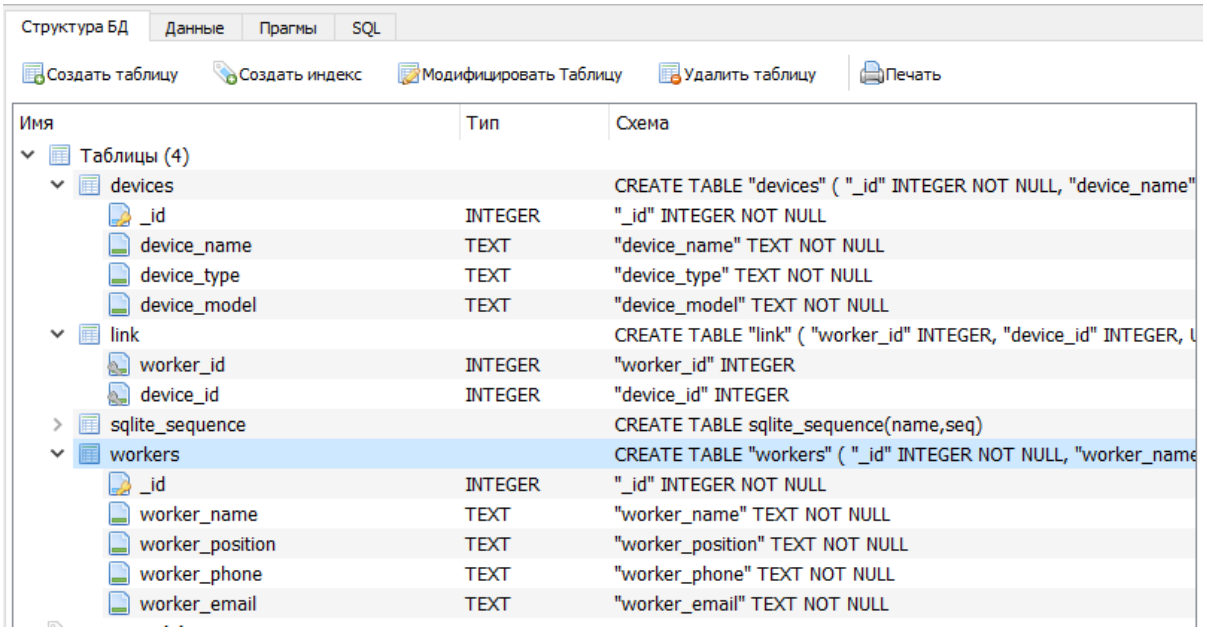
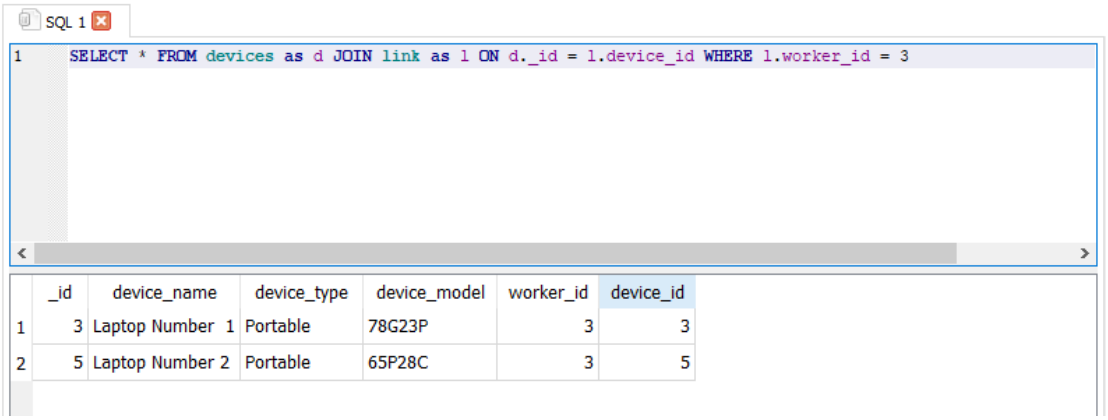


Рисунок 4.3.6 – Реалізація таблиць БД

Наступним кроком є наповнення таблиць даними для тестування та розробка запитів для перевірки коректності роботи БД. Для цього створюємо запит, який виводить інформацію про всі девайси пов’язані з конкретним працівником та тестуємо його працездатність(рис 4.3.7.).



	_id	device_name	device_type	device_model	worker_id	device_id
1	3	Laptop Number 1	Portable	78G23P	3	3
2	5	Laptop Number 2	Portable	65P28C	3	5

Рисунок 4.3.7 – Тестування запиту

Для більш зручного тестування, в БД була додана невелика кількість записів, таким чином є можливо вручну перевірити чи є результат виведений запитом коректним, в результаті перевірки було виявлено що так, результат є коректним, отже БД та запит працюють потрібним нам чином та є спроектовані відповідно до схеми представленої раніше, тому можливо переходити до етапу прямої розробки додатку в середовищі Android Studio, до якої експортувати запити на створення таблиць та на виконання дій з ними.

4.4. Розробка додатку

При роботі в середовищі Android Studio є два основних функціональних елемента, це java класи та layout елементи. Java клас відповідає за логіку додатку, прописує функціональну частину його роботи, а в контексті даного додатку використовується також для створення БД та установлення зв’язків з нею. Layout

елемент, це елемент розмітки, тобто це представлення дизайну конкретного екрану, або фрагменту, яке складається з множини елементів розміщених та закріплених на макеті таким чином, яким це необхідно для розробника, доступ до цих елементів отримується за допомогою функції FindViewById, після чого стає можливим впливати з класу на макет. В контексті даного додатку було розроблено 35 Java класів та 19 layout елементів, повний код додатку, включаючи лістинг цих класів та елементів, а також лістинг файлу маніфесту, наведений в додатку Б.

В даному розділі розберемо основні моменти розробки додатку, такі як:

- Екран заставки
- Логіка авторизаціїта
- Створення БД
- Модель працівника
- Функція читання всіх працівників з БД
- Функція читання девайсів привязаних до конкретного працівника з БД

Для розробки екрана заставки, спочатку розробляється layout елемент, відповідно до дизайну розробленому в попередньому розділі (рис.4.4.1).

Рисунок 4.4.1 – Layout SplashScreen

Далі необхідно прописати логіку його роботи, а саме вказати час, на який заставка затримається на екрані, екран на який буде здійснено перехід після заставки, а також запустити ProgresBar елемент. Для цього був написаний код в класі Splash Activity (рис. 4.4.2).

```
package it_school.sumdu.edu.linkcontrol;

import ...

@SuppressLint("CustomSplashScreen")
public class SplashActivity extends AppCompatActivity{

    ProgressBar pb;
    int counter = 0;

    protected void onCreate(Bundle savedInstanceState) {
        new Handler().postDelayed(() -> {
            Intent i = new Intent( packageContext: SplashActivity.this, Authorization.class);
            startActivity(i);
            finish();
        }, delayMillis: 3*1000);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);

        progress();
    }

    public void progress()
    {
        pb = findViewById(R.id.progressBar);

        final Timer t = new Timer();
        TimerTask tt = () -> {
                counter++;
                pb.setProgress(counter);

                if(counter == 100)
                    t.cancel();
        };
        t.schedule(tt, delay: 0, period: 27);
    }
}
```

Рисунок 4.4.2 – Код SplashActivity

Як ми бачимо з рисунку наведеного вище, після затримки, в 3 секунди, заставка змінюється на екран Авторизація, а також за ці три секунди, за допомогою функції progress() здійснюється робота елементу типу progressbar.

Наступним кроком є огляд авторизації в рамках даного проекту, цей процес виглядає таким чином, користувач потрапляє на екран з короткою інструкцією та кнопкою авторизації(рис. 4.4.3), при натисканні на цю кнопку, відкривається діалогове вікно(рис. 4.4.4), з полями в які необхідно ввести інформацію користувача, та двома кнопками, одна з яких перевіряє інформацію та в випадку

відповідності дає доступ до додатку, а інша скасовує операцію і закриває діалогове вікно.

Рисунок 4.4.3 – Layout авторизації

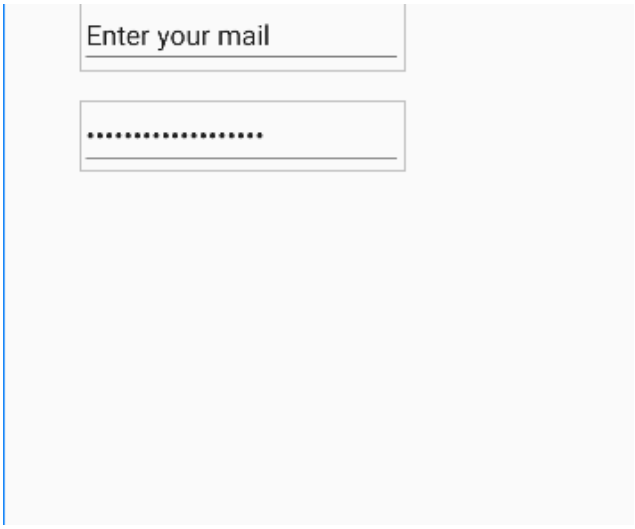


Рисунок 4.4.4 – Layout діалогового вікна

Логіка для цих двох layout прописана в класі Authorization (рис. 4.4.5-4.4.6).

```
public class Authorization extends AppCompatActivity
{
    FirebaseAuth auth;
    FirebaseDatabase db;
    DatabaseReference users;

    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auth);
        auth = FirebaseAuth.getInstance();
        db = FirebaseDatabase.getInstance();
        users = db.getReference("Users");


    }

    public void Sign(View view) {
        AlertDialog.Builder dialog = new AlertDialog.Builder( context: this);
        dialog.setTitle("Authorization");
        dialog.setMessage("Enter your details to log into your account");

        LayoutInflater inflater = LayoutInflater.from(this);
        View sign_in_window = inflater.inflate(R.layout.authorization, root: null);
        dialog.setView(sign_in_window);

        final EditText mail = sign_in_window.findViewById(R.id.mailFieldAuth);
        final EditText pass = sign_in_window.findViewById(R.id.passwordFieldAuth);

        dialog.setNegativeButton("Cancel", (dialogInterface, i) -> dialogInterface.dismiss());

        dialog.setPositiveButton("Authorization", (dialogInterface, i) -> {

            if(TextUtils.isEmpty(mail.getText().toString())||pass.getText().toString().length()<5)
            {
                Toast toast = Toast.makeText(getApplicationContext(),
                        "Incorrect filling of fields", Toast.LENGTH_SHORT);
                toast.show();
```

Рисунок 4.4.5 – Код класу авторизації, частина перша

```
                return;
            }
            auth.signInWithEmailAndPassword(mail.getText().toString(),pass.getText().toString())
                    .addOnSuccessListener(authResult -> {
                        Intent intent = new Intent( packageContext: this,WorkerListActivity.class);
                        startActivity(intent);
                    }).addOnFailureListener(e -> {
                        Toast toast = Toast.makeText(getApplicationContext(),
                                ""Error: "" + e.getMessage(), Toast.LENGTH_SHORT);
                        toast.show();
                    });
        });


        dialog.show();
    }
}
```

Рисунок 4.4.6 – Код класу авторизації, частина друга

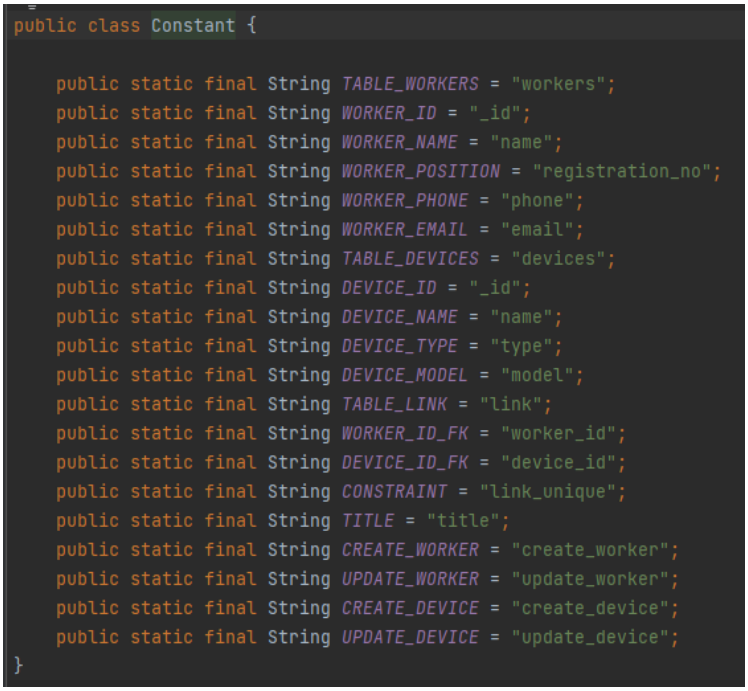
Як ми бачимо, авторизація відбувається в методі Sign, цей метод включає в себе виклик діалогового вікна, макету для нього, а також подальшу звірку даних введених користувачем, з даними, які є збережені в Firebase.

Після того, як користувач здійснив авторизацію, він потрапляє до основного функціоналу додатку, та бачить список з користувачами виведеними з БД, як було сказано в попередньому підрозділі, ця БД створюється напряму в класі Android Studio DatabaseHelper, за допомогою запитів (рис. 4.4.7) сформованих в SQLite DB Browser.

```
blic void onCreate(SQLiteDatabase sqLiteDatabase) {

String CREATE_WORKER_TABLE = "CREATE TABLE " + TABLE_WORKERS + "("
        + WORKER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + WORKER_NAME + " TEXT NOT NULL, "
        + WORKER_POSITION + " TEXT NOT NULL, "
        + WORKER_PHONE + " TEXT, " //nullable
        + WORKER_EMAIL + " TEXT " //nullable
        + ")";

String CREATE_DEVICES_TABLE = "CREATE TABLE " + TABLE_DEVICES + "("
        + DEVICE_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        + DEVICE_NAME + " TEXT NOT NULL, "
        + DEVICE_TYPE + " TEXT NOT NULL , "
        + DEVICE_MODEL + " TEXT NOT NULL" //nullable
        + ")";

String CREATE_LINK_TABLE = "CREATE TABLE " + TABLE_LINK + "("
        + WORKER_ID_FK + " INTEGER NOT NULL, "
        + DEVICE_ID_FK + " INTEGER NOT NULL, "
        + "FOREIGN KEY (" + WORKER_ID_FK + ") REFERENCES " + TABLE_WORKERS + "(" + WORKER_ID + ") " +
        "ON UPDATE CASCADE ON DELETE CASCADE, "
        + "FOREIGN KEY (" + DEVICE_ID_FK + ") REFERENCES " + TABLE_DEVICES + "(" + DEVICE_ID + ") " +
        "ON UPDATE CASCADE ON DELETE CASCADE, "
        + "CONSTRAINT " + CONSTRAINT + " UNIQUE (" + WORKER_ID_FK + "," + DEVICE_ID_FK + ")"
        + ")";

sqLiteDatabase.execSQL(CREATE_WORKER_TABLE);
sqLiteDatabase.execSQL(CREATE_DEVICES_TABLE);
sqLiteDatabase.execSQL(CREATE_LINK_TABLE);
```

Рисунок 4.4.7 – Створення таблиць

Значення для назв таблиць, та їх колонок, зберігаються в окремому класі Constants (рис. 4.4.8),це зроблено для того, щоб не додавати багато зайвих змінних в DatabaseHelper та в подальшому зробити зручнішим механізм звертання до цих таблиць в рамках решти частини коду.



```
public class Constant {

    public static final String TABLE_WORKERS = "workers";
    public static final String WORKER_ID = "_id";
    public static final String WORKER_NAME = "name";
    public static final String WORKER_POSITION = "registration_no";
    public static final String WORKER_PHONE = "phone";
    public static final String WORKER_EMAIL = "email";
    public static final String TABLE_DEVICES = "devices";
    public static final String DEVICE_ID = "_id";
    public static final String DEVICE_NAME = "name";
    public static final String DEVICE_TYPE = "type";
    public static final String DEVICE_MODEL = "model";
    public static final String TABLE_LINK = "link";
    public static final String WORKER_ID_FK = "worker_id";
    public static final String DEVICE_ID_FK = "device_id";
    public static final String CONSTRAINT = "link_unique";
    public static final String TITLE = "title";
    public static final String CREATE_WORKER = "create_worker";
    public static final String UPDATE_WORKER = "update_worker";
    public static final String CREATE_DEVICE = "create_device";
    public static final String UPDATE_DEVICE = "update_device";
}
```

Рисунок 4.4.8. – Клас Constants

Також створюються моделі для елементів кожної з таблиць, які дають можливість користувачу встановлювати нові значення, або отримувати значення, які є в наявності в БД. Наприклад модель працівника (рис. 4.4.9)

```java
package it_school.sumdu.edu.linkcontrol.model;

public class Worker {
    private int id;
    private String name;
    private String position;
    private String phone;
    private String email;

    public Worker(int id, String name, String position, String phone, String email) {
        this.id = id;
        this.name = name;
        this.position = position;
        this.phone = phone;
        this.email = email;
    }

    public int getId() { return id; }

    public void setId(int id) { this.id = id; }

    public String getName() { return name; }

    public void setName(String name) { this.name = name; }

    public String getPosition() { return position; }

    public void setPosition(String position) { this.position = position; }

    public String getPhone() { return phone; }

    public void setPhone(String phone) { this.phone = phone; }

    public String getEmail() { return email; }

    public void setEmail(String email) { this.email = email; }
}
```

Рисунок 4.4.9 - Модель працівника

Такі моделі, відповідно створюються для кожної з наявних таблиць. Далі створюється функція readAllWorkers (рис. 4.4.10), яка дозволяє читати інформацію про всіх працівників з БД та додає її в ArrayList, а в подальшому, за допомогою WorkerListActivity, з підтримкою WorkerListAdapter та WorkerListViewHolder, інформація з цього ArrayList виводиться вдоситься в відповідні поля CardView та в подальшому ці елементи CardView(рис. 4.4.11.) додаються в RecycleView і показується користувачеві.

```
public void readAllWorker(Response<List<Worker>> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();

    List<Worker> workerList = new ArrayList<>();

    Cursor cursor = null;
    try {
        cursor = sqLiteDatabase.query(TABLE_WORKERS, columns: null, selection: null, selectionArgs: null,
                groupBy: null, having: null, orderBy: null);

        if(cursor!=null && cursor.moveToFirst()){
            do {
                Worker worker = getWorkerFromCursor(cursor);
                workerList.add(worker);
            } while (cursor.moveToNext());

            response.onSuccess(workerList);
        } else
            response.onFailure("Worker list is empty");

    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
        if(cursor!=null)
            cursor.close();
    }
}
```

Рисунок 4.4.10 – Функція readAllWorkers

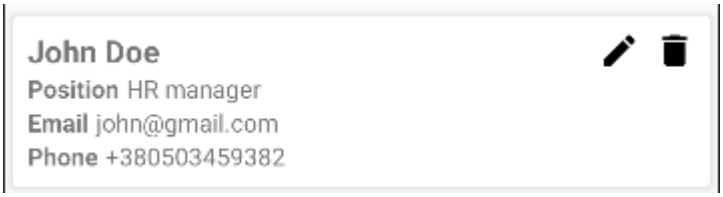


Рисунок 4.4.11 – Layout item_worker_card_view

Також в даному додатку існує функціонал, який дозволяє продивитись інформацію про працівника, та девайси, які є повязані з ним, для цього, кожен запис в RecycleView, вкладається в форматі CardView(рис. 4.4.11.), таким чином існує можливість взаємодії з ним, і користувач, після натискання на конкретного працівника, потрапляє до активності WorkerInfo(рис. 4.4.12.), де окрім інформації цього користувача, наведені девайси, які є пов’язані з ним, це реалізується за допомогою функції readAllLinkDeviceByWorkerId(рис.4.4.13.), в рамках цієї

функції є можливим бачити запит, за допомогою якого проводилось тестування працездатності моделі БД, створеної в SQLite Database Browser, відмінність полягає в тому, що в запиті для тестування, змінна WorkerID обиралась вручну, та була рівна 3, а в контексті даної функції, ця змінна є динамічною, та додається в запит залежно від елементу, на який натиснув користувач, тобто, якщо користувач натиснув на на працівника, _id якого рівне 5, то запит буде виконуватись з workerId==5.

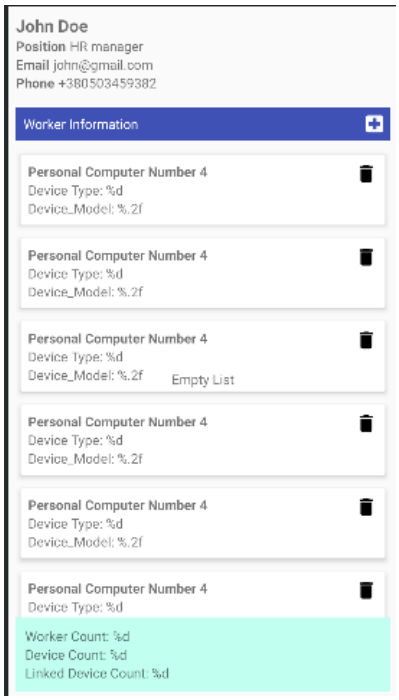
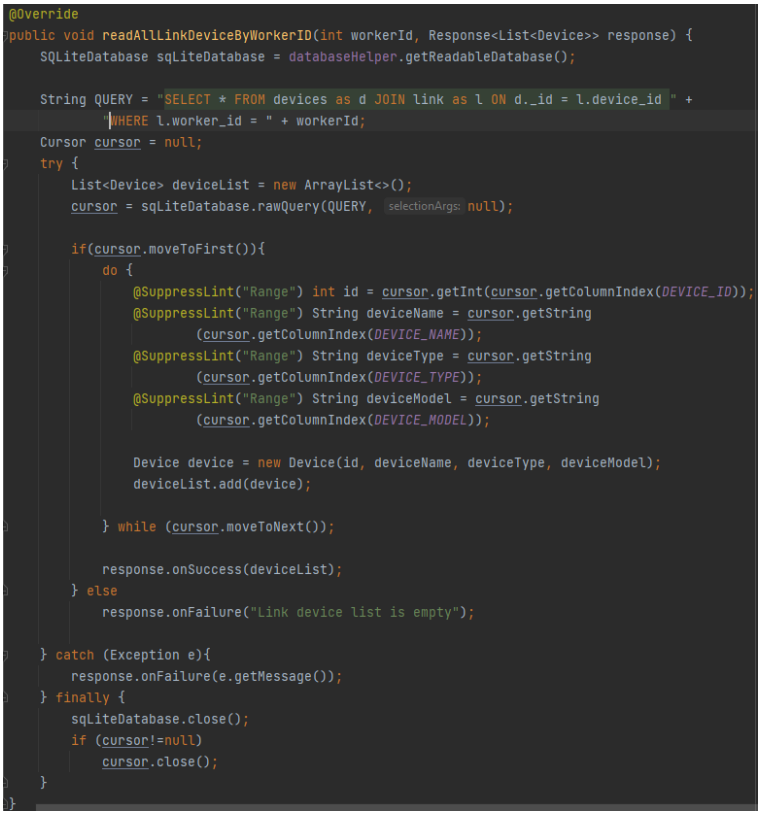


Рисунок 4.4.12 – Layout activity_worker_info



```
@Override
public void readAllLinkDeviceByWorkerID(int workerId, Response<List<Device>> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();

    String QUERY = "SELECT * FROM devices as d JOIN link as l ON d._id = l.device_id " +
            "WHERE l.worker_id = " + workerId;
    Cursor cursor = null;
    try {
        List<Device> deviceList = new ArrayList<>();
        cursor = sqLiteDatabase.rawQuery(QUERY, selectionArgs: null);

        if(cursor.moveToFirst()){
            do {
                @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(DEVICE_ID));
                @SuppressLint("Range") String deviceName = cursor.getString
                        (cursor.getColumnIndex(DEVICE_NAME));
                @SuppressLint("Range") String deviceType = cursor.getString
                        (cursor.getColumnIndex(DEVICE_TYPE));
                @SuppressLint("Range") String deviceModel = cursor.getString
                        (cursor.getColumnIndex(DEVICE_MODEL));

                Device device = new Device(id, deviceName, deviceType, deviceModel);
                deviceList.add(device);

            } while (cursor.moveToNext());

            response.onSuccess(deviceList);
        } else
            response.onFailure("Link device list is empty");

    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
        if (cursor!=null)
            cursor.close();
    }
}
```

Рисунок 4.4.13 – Функція readAllLinkDeviceByWorkerId

Варто зауважити, що в рамках даного підрозділу представлена реалізація базової частини функціоналу, повний код наведений в додатку Б, в ньому представлені всі класи та layout елементи, які були розроблені в рамках виконання даного проекту.

Таким чином результатом реалізації, за моделюванням описаним в третьому розділі, став мобільний додаток, який дозволяє авторизованому користувачеві отримувати доступ до БД та вносити в неї дані про працівників та девайси на підприємстві, а також встановлювати зв’язки між ними, що в свою чергу дозволяє організувати контроль за об’єктами та суб’єктами інформаційної системи на підприємстві, та функціонально покриває етапи під номерами 2,3,4 в технології описаній в рамках підрозділу 4.1 [24][25] .

ВИСНОВОК

Результатом виконання роботи, стала інформаційна технологія проектування базової системи захисту підприємства, описана в підрозділі 4.1., яка представляє з себе список етапів, які не є ресурсозатратними, а також більшість з них може бути реалізована особою без спеціалізованих навичок в сфері інформаційних технологій. Таким чином, реалізація цих етапів, дозволить сформувати захист інформації елементарного рівня на підприємстві, що в свою чергу, значно зменшує шанс виникнення проблем інформаційної безпеки.

В ході виконання даної роботи, також було розроблено програмний продукт, а саме додаток на основі Android “Link control”, даний додаток дозволяє контролювати зв’язки між об’єктами та суб’єктами в інформаційній системі підприємства. Основою для визначення функціональних вимог до додатку стала об'єктно-орієнтована методологія проектування комплексних систем захисту інформації. Таким чином, використання додатку, дозволить користувачу контролювати інформаційну систему підприємства та приймати заходи для підвищення рівня інформаційної безпеки цього підприємства. Також, окрім обліку, в додаток є влаштована область для контролю задач, таким чином користувач зможе ставити задачі, після чого позначати виконані, та ті які ще потребують виконання. Інформацію додаток отримує з бази даних, в яку користувач при потребі може додавати нові записи, використовуючи графічний інтерфейс додатку.

Окрім додатку, який є практичною складовою даної роботи, також була створена теоретична частина, яка висвітлює основні положення щодо створення комплексних систем захисту інформації, а також надає інформацію щодо грамотного проектування систем такого роду, та основних аспектів, які є ключовими в питанні комплексних систем захисту інформації. Були розглянуті

основні критерії, відповідністю до яких проводиться оцінка якості системи захисту інформації, а також наведено приклад реальних загроз для кожного з цих критеріїв, разом з прикладами того, як цих загроз можливо уникнути, чи мінімізувати.

Отже підсумовуючи всю роботу є можливим виділити основні результати, а саме:

- Було сформульовано інформаційну технологію проектування базової системи захисту інформації
- Було розроблено додаток на базі Android
- Спроектована та наповнена даними БД
- Створено теоретичний огляд питання КСЗІ
- Розглянуто основні засоби та методи захисту інформації.

СПИСОК ЛІТЕРАТУРИ

1. Про затвердження Порядку проведення державної експертизи комплексних систем захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах Служби безпеки України : Наказ Служби безпеки України від 22.07.2019 р. № 1132. URL: https://zakon.rada.gov.ua/laws/show/z0910-19#Text (дата звернення: 05.10.2022).

2. Комплексна система захисту інформації (КСЗІ). [Electronic resource]. – Access mode: https://uss.gov.ua/service/kszi/(дата звернення: 05.10.2022).

3. Комплексні системи захисту інформації : навч. посіб. / Ю. Є. Яремчук та ін. Вінниця: Вид-во ВНТУ, 2018. 118 с.

4. Необхідність створення комплексної системи захисту інформації. [Electronic resource]. – Access mode: https://tzi.com.ua/neobxdnst-stvorennya-kompleksno-sistemi-zaxistu-nformacz-ksz.html (дата звернення: 05.10.2022).

5. Principles and order of developing complex information security systems in information and telecommunication systems / U.V. Zemlyanko, A.A. Zamula, A.A. Tkach, N.I. Litvinova, Y.A. Peresechanskaya // Applied Radio Electronics: Sci. Mag. – 2010. Vol. 9. № 3. – P. 460-469 (дата звернення: 05.10.2022).

6. ISO 27000 // Wikipedia. [Electronic resource]. – Access mode : https://en.wikipedia.org/wiki/ISO_27000 (дата звернення: 05.10.2022).

7. ДСТУ ISO/IEC 27001:2015. Інформаційні технології – Методи захисту – Системи управління інформаційною безпекою – Вимоги. На заміну ISO/IEC 27001:2005 ; чинний від 2019-11-01. Вид. офіц.

8. ДСТУ ISO/IEC 27005:2019. Інформаційні технології. Методи захисту. Управління ризиками інформаційної безпеки. На заміну ДСТУ ISO/IEC 27005:2015 ; чинний від 2019-11-01. Вид. офіц.

9. ДСТУ ISO/IEC 27003:2018. Інформаційні технології. Методи захисту. Системи керування інформаційною безпекою. Настанова. Чинний від 2018-10-01. Вид. офіц.

10. ДСТУ ISO/IEC 27004:2018. Інформаційні технології. Методи захисту. Системи керування інформаційною безпекою. Моніторинг, вимірювання, аналізування та оцінювання. Чинний від 2018-10-01. Вид. офіц.

11. Security information and event management (SIEM) [Electronic resource] Access mode: https://www.imperva.com/learn/application-security/siem/

12. Andrzejewski, K. (2020). Security information management systems. Management Sciences, 24(4). https://doi.org/10.15611/ms.2019.4.01 (дата звернення: 05.10.2022).

13. Про захист інформації в інформаційно-телекомунікаційних системах: Закон України від 05.07.1994 р. № 80/94-ВР: станом на 1 лип. 2022 р. URL: https://zakon.rada.gov.ua/laws/show/80/94-вр#Text (дата звернення: 09.10.2022).

14. Жураковський Б. Ю., Недашківський О. Л. Системазахисту інформації при передачі даних в радіоканалі. Кібербезпека: освіта, наука, техніка. 2022. No 3(15).

15. Перелік документів системи технічного захисту інформації (НД ТЗІ) [Electronic resource]. – Access mode: https://cip.gov.ua/ua/news/perelik-dokumentiv-sistemi-tekhnichnogo-zakhistu-informaciyi-nd-tzi (дата звернення: 05.10.2022).

16. Components of Mustonen-Ollila, E., Lehto, M., & Heikkonen, J. (2020). Components of defence strategies in society’s information environment: a case study based on the grounded theory. Security and Defence Quarterly, 28(1). https://doi.org/10.35467/sdq/118186 (дата звернення: 05.10.2022).

17. Делембовський, М., Шабала, Є., & Терентьєв, О. (2021). Аналіз методів та шляхів вирішення захисту інформації в інформаційно-телекомунікаційних системах. Грааль науки, https://doi.org/10.36074/grail-of-science.19.02.2021.051 (дата звернення: 05.10.2022).

18. Сідлецький, Є. В. Організаційні засоби захисту інформації в системах електронного документообігу / Є. В. Сідлецький ; наук. керівник Т. А. Петренко // Новітні технології у науковій діяльності і навчальному процесі: зб. тез Всеукр. наук.-практ. конф. студентів, аспірантів та молодих учених (м. Чернігів, 8-9 квіт. 2020 р.) : збірник тез доп. – Чернігів : НУ «Чернігівська політехніка», 2020. – С. 144-146.

19. Криптографічні засоби захисту інформації [Electronic resource]. – Access mode : https://studfile.net/preview/5462915/ (дата звернення: 05.10.2022).

20. Shcheblanin, Y., & Rabchun, D. (2018). Mathematical model of information security's theat agent. Cybersecurity: Education, Science, Technique, 1. https://doi.org/10.28925/2663-4023.2018.1.6372 (дата звернення: 05.10.2022).

21. S. Btoush, E., & M. Hammad, M. (2015). Generating ER Diagrams from Requirement Specifications Based On Natural Language Processing. International Journal of Database Theory and Application, 8(2). https://doi.org/10.14257/ijdta.2015.8.2.07 (дата звернення: 05.10.2022).

22. Prototyping for all. [Electronic resource].– Access mode: https://proto.io/

23. Icons for android studio. [Electronic resource].– Access mode: https://icons8.com/icons/set/android-studio (дата звернення: 05.10.2022).

24. R. Sharma, "Android Application Development," International Journal for Research in Applied Science and Engineering Technology, vol. 9, no. VI, 2021, doi: 10.22214/ijraset.2021.35425

25. A. Nugraha and F. Aminur Rahman, "Android Application Development of Student Learning Skills in Era Society 5.0," in Journal of Physics: Conference Series, 2021, vol. 1779, no. 1. doi: 10.1088/1742-6596/1779/1/012014.

26. Á. Gunnarsson and M. Johnson, "Application Lifecycle Management," in Pro Microsoft Power BI Administration, 2020. doi: 10.1007/978-1-4842-6567-3_5.

27. Meng, C., & Baier, H. (2019). Bring2lite: A Structural Concept and Tool for Forensic Data Analysis and Recovery of Deleted SQLite Records. Digital

Investigation, 29. https://doi.org/10.1016/j.diin.2019.04.017 (дата звернення: 08.10.2022).

28. Димова, Г. (2021). Аналіз методів оцінки ефективності систем фізичного захисту. Computer-integrated technologies: education, science, production, 45. https://doi.org/10.36910/6775-2524-0560-2021-45-02

29. Легенчук, С. Ф., Царук, І. М., & Назаренко, Т. П. (2021). Принципи захисту даних у системі обліку: управлінські аспекти. Економіка, Управління Та Адміністрування, 2(96). https://doi.org/10.26642/ema-2021-2(96)-61-69

30. Равлюк, В., & Ваврічен, О. (2021). Визначення рівня захищеності інформаційно-комунікаційної системи корпоративних мереж. Збірник Наукових Праць Національної Академії Державної Прикордонної Служби України. Серія: Військові Та Технічні Науки, 83(2). https://doi.org/10.32453/3.v83i2.576 (дата звернення: 05.10.2022).

31. Khawas, C., & Shah, P. (2018). Application of Firebase in Android App Development-A Study. International Journal of Computer Applications, 179(46). https://doi.org/10.5120/ijca2018917200 (дата звернення: 03.10.2022).

32. Degtyareva, L., Miroshnykova, M., & Voloshko, S. (2019). Аналіз структури системи захисту інформації. Системи Управління, Навігації Та Зв’язку. Збірник Наукових Праць, 2(54). https://doi.org/10.26906/sunz.2019.2.078 (дата звернення: 03.10.2022).

ДОДАТОК А

ПЛАНУВАННЯ РОБІТ

А.1 Ідентифікація мети ІТ-проекту

Основною метою даного проекту є надання простого та доступного пояснення цінності захисту інформації, незалежно він структури організації та її направленості, а також створення інструменту, який дозволить вести облік інформаційної системи та приймати рішення щодо її захисту для людини, яка не має спеціалізованої освіти.

Проект поділяється на дві частини, які представлені відповідно у розділах 1-2 та 3-4. Перша частина проекту надає основні теоретичні знання про захист інформації та засоби, які використовуються для побудови комплексних систем захисту інформації. Ознайомлення з першою частиною роботи дозволяє отримати уявлення про безпеку інформації, на які саме її аспекти варто звертати найбільше уваги. Основною орієнтацією проекту є невеликі підприємства, яка зазвичай нехтують інформаційною безпекою, таким чином інформація наведена в першій частині дає пояснення того, чому інформаційна безпека це не завжди складні високотехнологічні структури, які потребують маси ресурсів та часу на їх підтримання, а також освітлюють фактор того, що багато засобів, які нічого не коштують організації в перспективі можуть позбавити її від багатьох проблем. Окрім всього наведеного вище, також ознайомлення з першою частиною роботи дозволить користувачу проводити паралелі між представленим інструментарієм і прикладами, та своїм підприємством.

Друга частина роботи є більш практичною та висвітлює цикл створення програмного продукту. Основними складовими інформаційної системи на будь-

якому підприємстві є об’єкти, суб’єкти, а також їх взаємозв’язки. Таким чином розроблений програмний продукт дозволяє регулювати ці основні компоненти та вест їх облік, а вже на основі цього обліку приймати рішення щодо інформаційної безпеки підприємства.

Таким чином, в результаті використання двох частин роботи користувач має теоретичну базу, яка надає йому базисні знання щодо безпеки, а також технологічне рішення, яке дає йому зручний інструмент для аналізу інформаційної системи підприємства та прийняття подальших рішень.

А.2 Планування змісту структури робіт ІТ-проекту

Структура робіт виконаних в рамках реалізації проекту виглядає таким чином:

- Розробка концепту проекту;
- Пошук інформації;
- Систематизація інформації;
- Розроблення теоретичної частини роботи;
- Формування функціональних вимог до практичної частини;
- Створення діаграм IDF та варіантів використання;
- Опис схеми роботи БД та її представлення в форматі ER діаграми;
- Створення дизайну додатку;
- Вибір засобів для технічної реалізації додатку;
- Реалізація БД;
- Розробка додатку;

- Тестування додатку;
- Виправлення знайдених помилок;
- Оформлення практичної частини роботи.

А.3 Побудова календарного графіку виконання ІТ – проекту

Робота над проектом велась починаючи з 03.10.2022 до 10.12.2022, таким чином робота йшла протягом 10 тижнів.

Календарний графік виконання проекту представлений на діаграмі Ганта, яка представлена на рисунку А.3.

Рисунок А.3. – Календарний графік виконання проекту

А.4 Аналіз можливих ризиків проекту

Ризики, які виникають в ході реалізації проекту представлені на таблиці А.4

Таблиця А.4. – Ризики проекту

мер	Ризик	Вірогідність	Шкода
R1	Використання неактуальних даних	2	3
R2	Зміна функціональних вимог	2	2
R3	Інфраструктурні проблеми	4	4
R4	Втрата напрацювань через технічні неполадки	2	5
R5	Оновлення бібліотек	3	1
R6	Затирання даних	1	3

Для кращої візуалізації, створимо матрицю, ця матриця дає певну оцінку кожному ризику, відповідно до відношення Вірогідність – Шкода. Матриця представлена на рисунку А.4

	Вірогідність	5	5	10	15	20	25
R3		4	4	8	12	16	20
R5		3	3	6	9	12	15
R1,R2, R4		2	2	4	6	8	10
R6		1	1	2	3	4	5
			1	2	3	4	5
			Шкода				
			R5	R2	R1, R6	R3	R4

Рисунок А.4. - Матриця Вірогідність – Шкода

ДОДАТОК Б

ЛІСТИНГИ

Count.java

```
package it_school.sumdu.edu.linkcontrol.model;


public class Count {
    private long workerRow;
    private long deviceRow;
    private long linkRow;


    public Count(long studentRow, long subjectRow, long takenSubjectRow)
    {
        this.workerRow = studentRow;
        this.deviceRow = subjectRow;
        this.linkRow = takenSubjectRow;
    }
    public long getWorkerRow()
    {
        return workerRow;
    }
    public long getDeviceRow()
    {
        return deviceRow;
    }
    public long getLinkRow()
    {
        return linkRow;
    }

            }
```

Device.java

```
package it_school.sumdu.edu.linkcontrol.model;


public class Device {
    private int id;
    private String name;
    private String type;
    private String model;

    public Device(int id, String name, String type, String model) {
        this.id = id;
        this.name = name;
        this.type = type;
        this.model = model;
    }
    public int getId()
    {
        return id;
    }
    public void setId(int id)
    {
        this.id = id;
    }
    public String getName()
    {
        return name;
    }
    public void setName(String name)
```

```
    {
        this.name = name;
    }
    public String getType()
    {
        return type;
    }
    public void setType(String type)
    {
        this.type = type;
    }
    public String getModel()
    {
        return model;
    }
    public void setModel(String model)
    {
        this.model = model;
    }
}
```

Link.java

```
package it_school.sumdu.edu.linkcontrol.model;

public class Link extends Device{

    private boolean isTaken;
```

```
    public Link(int id, String name, String type, String model, boolean isTaken)
    {
        super(id, name, type, model);
        this.isTaken = isTaken;
    }
    public boolean isTaken()
    {
        return isTaken;
    }
}
```

User.java

```
package it_school.sumdu.edu.linkcontrol.model;


public class User {
    private String mail,name,password;


    public User(){ }
    public User(String mail,String name, String password){
        this.mail = mail;
        this.name = name;
        this.password = password;
    }
    public String getMail() {
        return mail;
    }
    public void setMail(String mail) {
        this.mail = mail;
```

```
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

Worker.java

```
package it_school.sumdu.edu.linkcontrol.model;

public class Worker {
    private int id;
    private String name;
    private String position;
    private String phone;
    private String email;

    public Worker(int id, String name, String position, String phone, String email) {
        this.id = id;
        this.name = name;
```

```
    this.position = position;
    this.phone = phone;
    this.email = email;
}
public int getId()
{
    return id;
}
public void setId(int id)
{
    this.id = id;
}
public String getName()
{
    return name;
}
public void setName(String name)
{
    this.name = name;
}
public String getPosition()
{
    return position;
}
public void setPosition(String position)
{
    this.position = position;
```

```
    }
    public String getPhone()
    {
        return phone;
    }
    public void setPhone(String phone)
    {
        this.phone = phone;
    }
    public String getEmail()
    {
        return email;
    }
    public void setEmail(String email)
    {
        this.email = email;
    }
}
```

App.java

```
package it_school.sumdu.edu.linkcontrol.stat;

import android.app.Application;
import android.content.Context;

public class App extends Application {
    public static Context context = null;
    @Override
```

```
    public void onCreate() {
        super.onCreate();
        context = getApplicationContext();
    }
}
```

Constant.java

```
package it_school.sumdu.edu.linkcontrol.stat;


public class Constant {


    public static final String TABLE_WORKERS = "workers";
    public static final String WORKER_ID = "_id";
    public static final String WORKER_NAME = "name";
    public static final String WORKER_POSITION = "registration_no";
    public static final String WORKER_PHONE = "phone";
    public static final String WORKER_EMAIL = "email";
    public static final String TABLE_DEVICES = "devices";
    public static final String DEVICE_ID = "_id";
    public static final String DEVICE_NAME = "name";
    public static final String DEVICE_TYPE = "type";
    public static final String DEVICE_MODEL = "model";
    public static final String TABLE_LINK = "link";
    public static final String WORKER_ID_FK = "worker_id";
    public static final String DEVICE_ID_FK = "device_id";
    public static final String CONSTRAINT = "link_unique";
    public static final String TITLE = "title";
    public static final String CREATE_WORKER = "create_worker";
```

```
    public static final String UPDATE_WORKER = "update_worker";
    public static final String CREATE_DEVICE = "create_device";
    public static final String UPDATE_DEVICE = "update_device";
}
```

Authorization.java

```
package it_school.sumdu.edu.linkcontrol;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.EditText;
import android.widget.Toast;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import java.util.Objects;
import it_school.sumdu.edu.linkcontrol.model.User;


public class Authorization extends AppCompatActivity
{
    FirebaseAuth auth;
    FirebaseDatabase db;
```

```
    DatabaseReference users;


    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_auth);
        auth = FirebaseAuth.getInstance();
        db = FirebaseDatabase.getInstance();
        users = db.getReference(getString(R.string.db_table_name));
    }
    public void Sign(View view) {
        AlertDialog.Builder dialog = new AlertDialog.Builder(this);
        dialog.setTitle(R.string.auth_title);
        dialog.setMessage(R.string.auth_msg);
        LayoutInflater inflater = LayoutInflater.from(this);
        View sign_in_window = inflater.inflate(R.layout.authorization,null);
        dialog.setView(sign_in_window);
        final EditText mail = sign_in_window.findViewById(R.id.mailFieldAuth);
        final EditText pass = sign_in_window.findViewById(R.id.passwordFieldAuth);
        dialog.setNegativeButton(R.string.cancel, (dialogInterface, i) -> dialogInterface.dismiss());
        dialog.setPositiveButton(R.string.auth, (dialogInterface, i) -> {
if(TextUtils.isEmpty(mail.getText().toString())||pass.getText().toString().length()<5)
            {
                Toast toast = Toast.makeText(getApplicationContext(),
                        R.string.usr_mistake, Toast.LENGTH_SHORT);
                toast.show();
                return;
```

```
        }
auth.signInWithEmailAndPassword(mail.getText().toString(),pass.getText().toString())
                .addOnSuccessListener(authResult -> {
                    Intent intent = new Intent(this,WorkerListActivity.class);
                    startActivity(intent);
                }).addOnFailureListener(e -> {
                    Toast toast = Toast.makeText(getApplicationContext(),
                            getString(R.string.error_toast) + e.getMessage(), Toast.LENGTH_SHORT);
                    toast.show();
                });
        });
        dialog.show();
    }
}
```

ContactRequest.java

```
package it_school.sumdu.edu.linkcontrol;

import it_school.sumdu.edu.linkcontrol.model.Worker;
import it_school.sumdu.edu.linkcontrol.model.Device;
import it_school.sumdu.edu.linkcontrol.model.Count;
import it_school.sumdu.edu.linkcontrol.model.Link;
import java.util.List;

public class ContactRequest {

    public interface WorkerRequest {
        void createWorker(Worker worker, Response<Boolean> response);
```

```
        void readWorker(int workerId, Response<Worker> response);
        void readAllWorker(Response<List<Worker>> response);
        void updateWorker(Worker worker, Response<Boolean> response);
        void deleteWorker(int workerId, Response<Boolean> response);
    }
    public interface DeviceRequest {
        void createDevice(Device device, Response<Boolean> response);
        void readAllDevice(Response<List<Device>> response);
        void updateDevice(Device device, Response<Boolean> response);
        void deleteDevice(int deviceId, Response<Boolean> response);
    }
    public interface LinkRequest {
        void createLink(int workerId, int deviceId, Response<Boolean> response);
        void readAllLinkDeviceByWorkerID(int workerId, Response<List<Device>> response);
        void readAllLinkDevice(int workerId, Response<List<Link>> response);
        void deleteLinkDevice(int workerId, int deviceId, Response<Boolean> response);
    }
    public interface CountRequest {
        void getCount(Response<Count> response);
    }
}
```

CountImplementation.java

```
package it_school.sumdu.edu.linkcontrol;

import android.database.DatabaseUtils;
import android.database.sqlite.SQLiteDatabase;
import it_school.sumdu.edu.linkcontrol.model.Count;
```

```
import it_school.sumdu.edu.linkcontrol.stat.Constant;


public class CountImplementation implements ContactRequest.CountRequest {
    @Override
    public void getCount(Response<Count> response) {
        DatabaseHelper databaseHelper = DatabaseHelper.getInstance();
        try (SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase()) {
            long workerCount = DatabaseUtils.queryNumEntries(sqLiteDatabase, Constant.TABLE_WORKERS);
            long deviceCount = DatabaseUtils.queryNumEntries(sqLiteDatabase, Constant.TABLE_DEVICES);
            long linkCount = DatabaseUtils.queryNumEntries(sqLiteDatabase, Constant.TABLE_LINK);
            Count count = new Count(workerCount, deviceCount, linkCount);
            response.onSuccess(count);
        } catch (Exception e) {
            response.onFailure(e.getMessage());
        }
    }
}
```

DatabaseHelper.java

```
package it_school.sumdu.edu.linkcontrol;


import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import it_school.sumdu.edu.linkcontrol.stat.App;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;


public class DatabaseHelper extends SQLiteOpenHelper {
```

```
private static final String DATABASE_NAME = "linkControlDB";

private static final int DATABASE_VERSION = 1;

private static volatile DatabaseHelper databaseHelper;


private DatabaseHelper() {
   super(App.context, DATABASE_NAME, null, DATABASE_VERSION);
}
public static DatabaseHelper getInstance() {
   if (databaseHelper == null) {
      synchronized (DatabaseHelper.class){
         if (databaseHelper == null)
            databaseHelper = new DatabaseHelper();
      }
   }
   return databaseHelper;
}
@Override
public void onCreate(SQLiteDatabase sqLiteDatabase) {
   String CREATE_WORKER_TABLE = "CREATE TABLE " + TABLE_WORKERS + "("
         + WORKER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
         + WORKER_NAME + " TEXT NOT NULL, "
         + WORKER_POSITION + " TEXT NOT NULL, "
         + WORKER_PHONE + " TEXT, "
         + WORKER_EMAIL + " TEXT "
         + ")";
   String CREATE_DEVICES_TABLE = "CREATE TABLE " + TABLE_DEVICES + "("
         + DEVICE_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
```

```
            + DEVICE_NAME + " TEXT NOT NULL, "
            + DEVICE_TYPE + " TEXT NOT NULL , "
            + DEVICE_MODEL + " TEXT NOT NULL"
            + ")";
        String CREATE_LINK_TABLE = "CREATE TABLE " + TABLE_LINK + "("
            + WORKER_ID_FK + " INTEGER NOT NULL, "
            + DEVICE_ID_FK + " INTEGER NOT NULL, "
            + "FOREIGN KEY (" + WORKER_ID_FK + ") REFERENCES " + TABLE_WORKERS + "(" + WORKER_ID
+ ") " +
            "ON UPDATE CASCADE ON DELETE CASCADE, "
            + "FOREIGN KEY (" + DEVICE_ID_FK + ") REFERENCES " + TABLE_DEVICES + "(" + DEVICE_ID + ")
" +
            "ON UPDATE CASCADE ON DELETE CASCADE, "
            + "CONSTRAINT " + CONSTRAINT + " UNIQUE (" + WORKER_ID_FK + "," + DEVICE_ID_FK + ")"
            + ")";
        sqLiteDatabase.execSQL(CREATE_WORKER_TABLE);
        sqLiteDatabase.execSQL(CREATE_DEVICES_TABLE);
        sqLiteDatabase.execSQL(CREATE_LINK_TABLE);
    }
    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_WORKERS);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_DEVICES);
        sqLiteDatabase.execSQL("DROP TABLE IF EXISTS " + TABLE_LINK);

        onCreate(sqLiteDatabase);
    }
    @Override
```

```
    public void onOpen(SQLiteDatabase db) {
        super.onOpen(db);
        db.execSQL("PRAGMA foreign_keys=ON;");
    }
}
```

DeviceAddActivity.java

```
package it_school.sumdu.edu.linkcontrol;

import android.annotation.SuppressLint;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;
import it_school.sumdu.edu.linkcontrol.model.Link;
import java.util.ArrayList;
import java.util.List;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.WORKER_ID;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

public class DeviceAddActivity extends AppCompatActivity {
    private RecyclerView recyclerView;
    private TextView noDataFoundTextView;
    private final List<Link> linkList = new ArrayList<>();
    private DeviceAddListAdapter adapter;
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_link);
        getWindow().setLayout(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);
        recyclerView = findViewById(R.id.recyclerView);
        noDataFoundTextView = findViewById(R.id.noDataFoundTextView);
        int workerId = getIntent().getIntExtra(WORKER_ID, -1);
        adapter = new DeviceAddListAdapter(this, workerId, linkList);
        recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false));
        recyclerView.setAdapter(adapter);
        ContactRequest.LinkRequest request = new LinkImplementation();
        request.readAllLinkDevice(workerId, new Response<List<Link>>() {
            @SuppressLint("NotifyDataSetChanged")
            @Override
            public void onSuccess(List<Link> data) {
                recyclerView.setVisibility(View.VISIBLE);
                noDataFoundTextView.setVisibility(View.GONE);
                linkList.clear();
                linkList.addAll(data);
                adapter.notifyDataSetChanged();
            }
            @Override
            public void onFailure(String message) {
                recyclerView.setVisibility(View.GONE);
                noDataFoundTextView.setVisibility(View.VISIBLE);
            }
        });
```

```
    }
    public void closeActivity(View view) {
        finish();
    }
}
```

DeviceAddListAdapter.java

```
package it_school.sumdu.edu.linkcontrol;


import android.content.Context;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CompoundButton;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Link;
import java.util.List;


public class DeviceAddListAdapter extends RecyclerView.Adapter<DeviceAddViewHolder> {
    private final Context context;
    private final int workerId;
    private final List<Link> linkList;
    public DeviceAddListAdapter(Context context, int workerId, List<Link> linkList) {
        this.context = context;
        this.workerId = workerId;
        this.linkList = linkList;
```

```
}
@NonNull
@Override
public DeviceAddViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(context).inflate(R.layout.item_link_card_view, parent, false);
    return new DeviceAddViewHolder(view);
}
@Override
public void onBindViewHolder(@NonNull DeviceAddViewHolder holder,  int position) {
    final Link device = linkList.get(position);
    holder.deviceNameTextView.setText(device.getName());
    holder.deviceTypeTextView.setText(device.getType());
    holder.deviceModelTextView.setText(device.getModel());
    holder.checkBox.setChecked(device.isTaken());
    holder.checkBox.setOnCheckedChangeListener((buttonView, isChecked) -> {
        if (isChecked)
            linkDevice(device.getId());
        else
            removeLinkDevice(device.getId());
    });
}
@Override
public int getItemCount() {
    return linkList.size();
}
private void linkDevice(int deviceId) {
    ContactRequest.LinkRequest request = new LinkImplementation();
```

```
        request.createLink(workerId, deviceId, new Response<Boolean>() {
            @Override
            public void onSuccess(Boolean data) {
            }
            @Override
            public void onFailure(String message) {
                Toast.makeText(context, message, Toast.LENGTH_LONG).show();
            }
        });
    }
    private void removeLinkDevice(int deviceId) {
        ContactRequest.LinkRequest request = new LinkImplementation();
        request.deleteLinkDevice(workerId, deviceId, new Response<Boolean>() {
            @Override
            public void onSuccess(Boolean data) {
            }
            @Override
            public void onFailure(String message) {
                Toast.makeText(context, message, Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

DeviceAddViewHolder.java

```
package it_school.sumdu.edu.linkcontrol;


import androidx.recyclerview.widget.RecyclerView;
```

```
import android.view.View;

import android.widget.CheckBox;

import android.widget.TextView;

import it_school.sumdu.edu.linkcontrol.R;


public class DeviceAddViewHolder extends RecyclerView.ViewHolder {


    CheckBox checkBox;

    TextView deviceNameTextView;

    TextView deviceTypeTextView;

    TextView deviceModelTextView;


    public DeviceAddViewHolder(View itemView) {

        super(itemView);

        checkBox = itemView.findViewById(R.id.checkbox);

        deviceNameTextView = itemView.findViewById(R.id.deviceNameTextView);

        deviceTypeTextView = itemView.findViewById(R.id.deviceTypeTextView);

        deviceModelTextView = itemView.findViewById(R.id.deviceModelTextView);

    }

}
```

DeviceCreate.java

```
package it_school.sumdu.edu.linkcontrol;


import android.app.Dialog;

import androidx.fragment.app.DialogFragment;

import android.os.Bundle;

import androidx.annotation.Nullable;
```

```
import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.EditText;

import android.widget.Toast;

import it_school.sumdu.edu.linkcontrol.model.Device;

import it_school.sumdu.edu.linkcontrol.stat.Constant;


public class DeviceCreate extends DialogFragment {


    private EditText deviceNameEditText;

    private EditText deviceTypeEditText;

    private EditText deviceModelEditText;

    private Button createButton;

    private Button cancelButton;

    private static DeviceListener deviceListener;

    public DeviceCreate() {

    }

    public static DeviceCreate newInstance(String title, DeviceListener listener){

        deviceListener = listener;

        DeviceCreate deviceCreate = new DeviceCreate();

        Bundle args = new Bundle();

        args.putString("title", title);

        deviceCreate.setArguments(args);

        deviceCreate.setStyle(DialogFragment.STYLE_NORMAL, R.style.CustomDialog);

        return deviceCreate;
```

```
}
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_device_create, container, false);
    deviceNameEditText = view.findViewById(R.id.deviceNameEditText);
    deviceTypeEditText = view.findViewById(R.id.deviceTypeEditText);
    deviceModelEditText = view.findViewById(R.id.deviceModelEditText);
    createButton = view.findViewById(R.id.createButton);
    cancelButton = view.findViewById(R.id.cancelButton);
    String title = getArguments().getString(Constant.TITLE);
    getDialog().setTitle(title);
    createButton.setOnClickListener(view1 -> {
        String deviceName = deviceNameEditText.getText().toString();
        String deviceType = deviceTypeEditText.getText().toString();
        String deviceModel = deviceModelEditText.getText().toString();
        final Device device = new Device(-1, deviceName, deviceType, deviceModel);
        ContactRequest.DeviceRequest request = new DeviceImplementation();
        request.createDevice(device, new Response<Boolean>() {
            @Override
            public void onSuccess(Boolean data) {
                getDialog().dismiss();
                deviceListener.onDeviceListUpdate(true);
                Toast.makeText(getContext(), "Device created successfully", Toast.LENGTH_LONG).show();
            }
            @Override
            public void onFailure(String message) {
```

```
                deviceListener.onDeviceListUpdate(false);
                Toast.makeText(getContext(), message, Toast.LENGTH_LONG).show();
            }
        });
    });
    cancelButton.setOnClickListener(view12 -> getDialog().dismiss());
    return view;
  }
  @Override
  public void onStart() {
    super.onStart();
    Dialog dialog = getDialog();
    if (dialog != null) {
      int width = ViewGroup.LayoutParams.MATCH_PARENT;
      int height = ViewGroup.LayoutParams.WRAP_CONTENT;
      dialog.getWindow().setLayout(width, height);
    }
  }
}
```

DeviceImplementation.java

```
package it_school.sumdu.edu.linkcontrol;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
```

```
import it_school.sumdu.edu.linkcontrol.model.Device;
import java.util.ArrayList;
import java.util.List;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.DEVICE_ID;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.DEVICE_NAME;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.DEVICE_TYPE;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.DEVICE_MODEL;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.TABLE_WORKERS;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.TABLE_DEVICES;

public class DeviceImplementation implements ContactRequest.DeviceRequest {

    private final DatabaseHelper databaseHelper = DatabaseHelper.getInstance();
    @Override
    public void createDevice(Device device, Response<Boolean> response) {
        SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put(DEVICE_NAME, device.getName());
        contentValues.put(DEVICE_TYPE, device.getType());
        contentValues.put(DEVICE_MODEL, device.getModel());
        try {
            long id = sqLiteDatabase.insertOrThrow(TABLE_DEVICES, null, contentValues);
            if(id>0) {
                response.onSuccess(true);
            }
            else
                response.onFailure("Create device error");
```

```
    } catch (SQLiteException e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
    }
}
@Override
public void readAllDevice(Response<List<Device>> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();
    List<Device> deviceList = new ArrayList<>();
    Cursor cursor = null;
    try {
        cursor = sqLiteDatabase.query(TABLE_DEVICES, null, null, null, null, null, null);
        if(cursor!=null && cursor.moveToFirst()){
            do {
                Device device = getDeviceFromCursor(cursor);
                deviceList.add(device);
            } while (cursor.moveToNext());
            response.onSuccess(deviceList);
        } else
            response.onFailure("Device database is empty");
    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
        if (cursor!=null)
            cursor.close();
```

```
    }
}
@Override
public void updateDevice(Device device, Response<Boolean> response) {
    try (SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase()) {
        ContentValues contentValues = getContentValuesFromDevice(device);
        long rowCount = sqLiteDatabase.update(TABLE_DEVICES, contentValues,
                DEVICE_ID + " =? ", new String[]{String.valueOf(device.getId())});
        if (rowCount > 0)
            response.onSuccess(true);
        else
            response.onFailure("Update device error");
    } catch (Exception e) {
        response.onFailure(e.getMessage());
    }
}
@Override
public void deleteDevice(int deviceId, Response<Boolean> response) {
    try (SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase()) {
        long rowCount = sqLiteDatabase.delete(TABLE_DEVICES,
                DEVICE_ID + " =? ", new String[]{String.valueOf(deviceId)});
        if (rowCount > 0)
            response.onSuccess(true);
        else
            response.onFailure("Delete object list is empty");
    } catch (Exception e) {
        response.onFailure(e.getMessage());
```

```
        }
    }
    private Device getDeviceFromCursor(Cursor cursor) {
        @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(DEVICE_ID));
        @SuppressLint("Range") String deviceName = cursor.getString(cursor.getColumnIndex(DEVICE_NAME));
        @SuppressLint("Range") String deviceType = cursor.getString(cursor.getColumnIndex(DEVICE_TYPE));
        @SuppressLint("Range") String deviceModel = cursor.getString(cursor.getColumnIndex(DEVICE_MODEL));
        return new Device(id, deviceName, deviceType, deviceModel);
    }
    private ContentValues getContentValuesFromDevice(Device device) {
        ContentValues contentValues = new ContentValues();
        contentValues.put(DEVICE_NAME, device.getName());
        contentValues.put(DEVICE_TYPE, device.getType());
        contentValues.put(DEVICE_MODEL, device.getModel());
        return contentValues;
    }
}
```

DeviceListActivity.java

```
package it_school.sumdu.edu.linkcontrol;


import android.annotation.SuppressLint;
import android.os.Bundle;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.Toolbar;
```

```
import android.view.View;

import android.widget.TextView;

import it_school.sumdu.edu.linkcontrol.model.*;

import java.util.ArrayList;

import java.util.List;

import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;


public class DeviceListActivity extends AppCompatActivity implements DeviceListener {


    private RecyclerView recyclerView;

    private TextView noDataFoundTextView;

    private TextView workerCountTextView;

    private TextView deviceCountTextView;

    private TextView linkCountTextView;

    private FloatingActionButton fab;

    private List<Device> deviceList = new ArrayList<>();

    private DeviceListAdapter adapter;

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_device_list);

        Toolbar toolbar = findViewById(R.id.toolbar);

        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        getSupportActionBar().setDisplayShowHomeEnabled(true);

        initialization();

        adapter = new DeviceListAdapter(this, deviceList, this);
```

```
    recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false));
    recyclerView.setAdapter(adapter);
    showTableRowCount();
    showSubjectList();
    fab.setOnClickListener(view -> {
        DeviceCreate dialogFragment = DeviceCreate.newInstance("Create Device", DeviceListActivity.this);
        dialogFragment.show(getSupportFragmentManager(), CREATE_DEVICE);
    });
}
@Override
public void onDeviceListUpdate(boolean isUpdate) {
    if(isUpdate) {
        showTableRowCount();
        showSubjectList();
    }
}
@Override
public boolean onSupportNavigateUp() {
    finish();
    return super.onSupportNavigateUp();
}
private void showTableRowCount() {
    ContactRequest.CountRequest request = new CountImplementation();
    request.getCount(new Response<Count>() {
        @Override
        public void onSuccess(Count data) {
            workerCountTextView.setText(getString(R.string.worker_count, data.getWorkerRow()));
```

```
            deviceCountTextView.setText(getString(R.string.device_count, data.getDeviceRow()));
            linkCountTextView.setText(getString(R.string.link_count, data.getLinkRow()));
        }
        @Override
        public void onFailure(String message) {
            workerCountTextView.setText(getString(R.string.count_error));
            deviceCountTextView.setText(message);
            linkCountTextView.setText("");
        }
    });
}
private void showSubjectList(){
    ContactRequest.DeviceRequest request = new DeviceImplementation();
    request.readAllDevice(new Response<List<Device>>() {
        @SuppressLint("NotifyDataSetChanged")
        @Override
        public void onSuccess(List<Device> data) {
            recyclerView.setVisibility(View.VISIBLE);
            noDataFoundTextView.setVisibility(View.GONE);
            deviceList.clear();
            deviceList.addAll(data);
            adapter.notifyDataSetChanged();
        }
        @Override
        public void onFailure(String message) {
            recyclerView.setVisibility(View.GONE);
            noDataFoundTextView.setVisibility(View.VISIBLE);
```

```
            }
        });
    }
    private void initialization(){
        recyclerView = findViewById(R.id.recyclerView);
        noDataFoundTextView = findViewById(R.id.noDataFoundTextView);
        workerCountTextView = findViewById(R.id.workerCount);
        deviceCountTextView = findViewById(R.id.devicesCount);
        linkCountTextView = findViewById(R.id.linkCount);
        fab = findViewById(R.id.fab);
    }
}
```

DeviceListAdapter.java

```
package it_school.sumdu.edu.linkcontrol;


import android.content.Context;
import android.content.DialogInterface;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Device;
import java.util.List;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;
```

```
public class DeviceListAdapter extends RecyclerView.Adapter<DeviceViewHolder> {
    private final Context context;
    private final List<Device> deviceList;
    private final DeviceListener listener;
    public DeviceListAdapter(Context context, List<Device> deviceList, DeviceListener listener) {
        this.context = context;
        this.deviceList = deviceList;
        this.listener = listener;
    }
    @NonNull
    @Override
    public DeviceViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.item_device_card_view, parent, false);
        return new DeviceViewHolder(view);
    }
    @Override
    public void onBindViewHolder(@NonNull DeviceViewHolder holder, int position) {
        final Device device = deviceList.get(position);
        holder.deviceNameTextView.setText(device.getName());
        holder.deviceTypeTextView.setText(device.getType());
        holder.deviceModelTextView.setText(device.getModel());
        holder.editIcon.setOnClickListener(v -> {
            DeviceUpdate deviceUpdate = DeviceUpdate.newInstance(device, "Update Subject", isUpdate ->
listener.onDeviceListUpdate(isUpdate));
            deviceUpdate.show(((DeviceListActivity) context).getSupportFragmentManager(), UPDATE_DEVICE);
        });
        holder.deleteIcon.setOnClickListener(v -> showConfirmationDialog(device.getId()));
```

```
}
@Override
public int getItemCount() {
    return deviceList.size();
}
private void showConfirmationDialog(final int deviceId) {
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(context);
    alertDialogBuilder.setMessage("Are you sure, You wanted to delete this device?");
    alertDialogBuilder.setPositiveButton("Yes",
            (arg0, arg1) -> {
                ContactRequest.DeviceRequest request = new DeviceImplementation();
                request.deleteDevice(deviceId, new Response<Boolean>() {
                    @Override
                    public void onSuccess(Boolean data) {
                        listener.onDeviceListUpdate(data);
                        Toast.makeText(context, "Device is deleted successfully", Toast.LENGTH_SHORT).show();
                    }
                    @Override
                    public void onFailure(String message) {
                        Toast.makeText(context, message, Toast.LENGTH_LONG).show();
                    }
                });
            });
    alertDialogBuilder.setNegativeButton("No", (dialog, which) -> dialog.dismiss());
    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
}
```

```
}
```

DeviceViewHolder.java

```
package it_school.sumdu.edu.linkcontrol;


import androidx.recyclerview.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import it_school.sumdu.edu.linkcontrol.R;


public class DeviceViewHolder extends RecyclerView.ViewHolder {
    TextView deviceNameTextView;
    TextView deviceTypeTextView;
    TextView deviceModelTextView;
    ImageView editIcon;
    ImageView deleteIcon;


    public DeviceViewHolder(View itemView) {
        super(itemView);
        deviceNameTextView = itemView.findViewById(R.id.deviceNameTextView);
        deviceTypeTextView = itemView.findViewById(R.id.deviceTypeTextView);
        deviceModelTextView = itemView.findViewById(R.id.deviceModelTextView);
        editIcon = itemView.findViewById(R.id.editIcon);
        deleteIcon = itemView.findViewById(R.id.deleteIcon);
    }
}
```

DeviceListener.java

```
package it_school.sumdu.edu.linkcontrol;


public interface DeviceListener {
    void onDeviceListUpdate(boolean isUpdate);
}
```

DeviceUpdate.java

```
package it_school.sumdu.edu.linkcontrol;


import android.app.Dialog;
import android.os.Bundle;
import androidx.fragment.app.DialogFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Device;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.TITLE;
import java.util.Objects;


public class DeviceUpdate extends DialogFragment {
    private static DeviceListener deviceListener;
    private EditText deviceNameEditText;
    private EditText deviceTypeEditText;
    private EditText deviceModelEditText;
    private static Device device;
```

```
public DeviceUpdate() {
}
public static DeviceUpdate newInstance(Device dev, String title, DeviceListener listener){
    device = dev;
    deviceListener = listener;
    DeviceUpdate deviceUpdate = new DeviceUpdate();
    Bundle args = new Bundle();
    args.putString("title", title);
    deviceUpdate.setArguments(args);
    deviceUpdate.setStyle(DialogFragment.STYLE_NORMAL, R.style.CustomDialog);
    return deviceUpdate;
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_device_update, container, false);
    assert getArguments() != null;
    String title = getArguments().getString(TITLE);
    Objects.requireNonNull(getDialog()).setTitle(title);
    deviceNameEditText = view.findViewById(R.id.deviceNameEditText);
    deviceTypeEditText = view.findViewById(R.id.deviceTypeEditText);
    deviceModelEditText = view.findViewById(R.id.deviceModelEditText);
    Button updateButton = view.findViewById(R.id.updateButton);
    Button cancelButton = view.findViewById(R.id.cancelButton);
    deviceNameEditText.setText(device.getName());
    deviceTypeEditText.setText(device.getType());
    deviceModelEditText.setText(device.getModel());
```

```
    updateButton.setOnClickListener(view1 -> {
        String deviceName = deviceNameEditText.getText().toString();
        String deviceType = deviceTypeEditText.getText().toString();
        String deviceModel = deviceModelEditText.getText().toString();
        device.setName(deviceName);
        device.setType(deviceType);
        device.setModel(deviceModel);
        ContactRequest.DeviceRequest deviceRequest = new DeviceImplementation();
        deviceRequest.updateDevice(device, new Response<Boolean>() {
            @Override
            public void onSuccess(Boolean data) {
                Objects.requireNonNull(getDialog()).dismiss();
                deviceListener.onDeviceListUpdate(data);
                Toast.makeText(getContext(), "Device updated successfully", Toast.LENGTH_LONG).show();
            }
            @Override
            public void onFailure(String message) {
                deviceListener.onDeviceListUpdate(false);
                Toast.makeText(getContext(), message, Toast.LENGTH_LONG).show();
            }
        });
    });
    cancelButton.setOnClickListener(view12 -> getDialog().dismiss());
    return view;
}
@Override
public void onStart() {
```

```
        super.onStart();
        Dialog dialog = getDialog();
        if (dialog != null) {
            int width = ViewGroup.LayoutParams.MATCH_PARENT;
            int height = ViewGroup.LayoutParams.WRAP_CONTENT;
            dialog.getWindow().setLayout(width, height);
        }
    }
}
```

LinkActivity.java

```
package it_school.sumdu.edu.linkcontrol;


import android.annotation.SuppressLint;
import android.content.Intent;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Worker;
import it_school.sumdu.edu.linkcontrol.model.Device;
import it_school.sumdu.edu.linkcontrol.model.Count;
import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Objects;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;

public class LinkActivity extends AppCompatActivity implements LinkListener {
    private TextView nameTextView;
    private TextView positionTextView;
    private TextView emailTextView;
    private TextView phoneTextView;
    private ImageView actionAddDevice;
    private RecyclerView recyclerView;
    private TextView noDataFoundTextView;
    private TextView workerCountTextView;
    private TextView deviceCountTextView;
    private TextView linkCountTextView;

    private int workerId;
    private final List<Device> linkList = new ArrayList<>();
    private LinkListAdapter adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_worker_info);
        Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
        initialization();
        workerId = getIntent().getIntExtra(WORKER_ID, -1);
        adapter = new LinkListAdapter(this, workerId, linkList, this);
```

```
    recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false));
    recyclerView.setAdapter(adapter);
    showWorkerInfo();
    actionAddDevice.setOnClickListener(v -> {
        Intent intent = new Intent(LinkActivity.this, DeviceAddActivity.class);
        intent.putExtra(WORKER_ID, workerId);
        startActivity(intent);
    });
}
@Override
protected void onResume() {
    super.onResume();
    showTableRowCount();
    showLinkList();
}
@Override
public boolean onSupportNavigateUp() {
    finish();
    return true;
}
@Override
public void onLinkUpdated(boolean isUpdated) {
    showTableRowCount();
}
private void showWorkerInfo() {
    ContactRequest.WorkerRequest request = new WorkerImplementation();
    request.readWorker(workerId, new Response<Worker>() {
```

```
        @Override
        public void onSuccess(Worker worker) {
            nameTextView.setText(worker.getName());
            positionTextView.setText(worker.getPosition());
            emailTextView.setText(worker.getEmail());
            phoneTextView.setText(worker.getPhone());
        }
        @Override
        public void onFailure(String message) {
            showToast(message);
        }
    });
}
private void showLinkList() {
    ContactRequest.LinkRequest request = new LinkImplementation();
    request.readAllLinkDeviceByWorkerID(workerId, new Response<List<Device>>() {
        @SuppressLint("NotifyDataSetChanged")
        @Override
        public void onSuccess(List<Device> data) {
            recyclerView.setVisibility(View.VISIBLE);
            noDataFoundTextView.setVisibility(View.GONE);
            linkList.clear();
            linkList.addAll(data);
            adapter.notifyDataSetChanged();
        }
        @Override
        public void onFailure(String message) {
```

```
            recyclerView.setVisibility(View.GONE);
            noDataFoundTextView.setVisibility(View.VISIBLE);
            noDataFoundTextView.setText(message);
        }
    });
}
private void showTableRowCount() {
    ContactRequest.CountRequest request = new CountImplementation();
    request.getCount(new Response<Count>() {
        @Override
        public void onSuccess(Count data) {
            workerCountTextView.setText(getString(R.string.worker_count, data.getWorkerRow()));
            deviceCountTextView.setText(getString(R.string.device_count, data.getDeviceRow()));
            linkCountTextView.setText(getString(R.string.link_count, data.getLinkRow()));
        }
        @Override
        public void onFailure(String message) {
            workerCountTextView.setText(getString(R.string.count_error));
            deviceCountTextView.setText(message);
            linkCountTextView.setText("");
        }
    });
}
private void initialization() {
    nameTextView = findViewById(R.id.nameTextView);
    positionTextView = findViewById(R.id.workerPositionTextView);
    emailTextView = findViewById(R.id.emailTextView);
```

```
        phoneTextView = findViewById(R.id.phoneTextView);
        actionAddDevice= findViewById(R.id.action_add_device);
        recyclerView = findViewById(R.id.recyclerView);
        noDataFoundTextView = findViewById(R.id.noDataFoundTextView);
        workerCountTextView = findViewById(R.id.workerCount);
        deviceCountTextView = findViewById(R.id.devicesCount);
        linkCountTextView = findViewById(R.id.linkCount);
    }
    private void showToast(String message){
        Toast.makeText(this, message, Toast.LENGTH_LONG).show();
    }
}
```

LinkImplementation.java

```
package it_school.sumdu.edu.linkcontrol;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import it_school.sumdu.edu.linkcontrol.model.Device;
import it_school.sumdu.edu.linkcontrol.model.Link;
import java.util.ArrayList;
import java.util.List;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;

public class LinkImplementation implements ContactRequest.LinkRequest {
```

```
private final DatabaseHelper databaseHelper = DatabaseHelper.getInstance();
@Override
public void createLink(int workerId, int deviceId, Response<Boolean> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase();
    ContentValues contentValues = new ContentValues();
    contentValues.put(WORKER_ID_FK, workerId);
    contentValues.put(DEVICE_ID_FK, deviceId);
    try {
        long rowCount = sqLiteDatabase.insertOrThrow(TABLE_LINK, null, contentValues);
        if (rowCount>0)
            response.onSuccess(true);
        else
            response.onFailure("Device link error");
    } catch (SQLiteException e) {
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
    }
}
@Override
public void readAllLinkDeviceByWorkerID(int workerId, Response<List<Device>> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();
    String QUERY = "SELECT * FROM devices as d JOIN link as l ON d._id = l.device_id " +
            "WHERE l.worker_id = " + workerId;
    Cursor cursor = null;
    try {
        List<Device> deviceList = new ArrayList<>();
```

```
        cursor = sqLiteDatabase.rawQuery(QUERY, null);
        if(cursor.moveToFirst()){
            do {
                @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(DEVICE_ID));
                @SuppressLint("Range") String deviceName = cursor.getString
                        (cursor.getColumnIndex(DEVICE_NAME));
                @SuppressLint("Range") String deviceType = cursor.getString
                        (cursor.getColumnIndex(DEVICE_TYPE));
                @SuppressLint("Range") String deviceModel = cursor.getString
                        (cursor.getColumnIndex(DEVICE_MODEL));
                Device device = new Device(id, deviceName, deviceType, deviceModel);
                deviceList.add(device);
            } while (cursor.moveToNext());
            response.onSuccess(deviceList);
        } else
            response.onFailure("Link device list is empty");
    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
        if (cursor!=null)
            cursor.close();
    }
}
@SuppressLint("Range")
@Override
public void readAllLinkDevice(int workerId, Response<List<Link>> response) {
```

```
        SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();
        String QUERY = "SELECT d._id, d.name, d.type, d.model, l.worker_id " +
                "FROM devices as d LEFT JOIN link as l ON d._id = l.device_id " +
                "AND l.worker_id = " + workerId;
        Cursor cursor = null;
        try {
            List<Link> linkList = new ArrayList<>();
            cursor = sqLiteDatabase.rawQuery(QUERY, null);
            if(cursor.moveToFirst()){
                do {
                    @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(DEVICE_ID));
                    @SuppressLint("Range") String deviceName = cursor.getString(cursor.getColumnIndex(DEVICE_NAME));
                    @SuppressLint("Range") String deviceType = cursor.getString(cursor.getColumnIndex(DEVICE_TYPE));
                    @SuppressLint("Range") String deviceModel =
cursor.getString(cursor.getColumnIndex(DEVICE_MODEL));
                    boolean isTaken = cursor.getInt(cursor.getColumnIndex(WORKER_ID_FK)) > 0;
                    Link link = new Link(id, deviceName, deviceType, deviceModel, isTaken);
                    linkList.add(link);
                } while (cursor.moveToNext());
                response.onSuccess(linkList);
            } else
                response.onFailure("Link list is empty");
        } catch (Exception e){
            response.onFailure(e.getMessage());
        } finally {
            sqLiteDatabase.close();
            if (cursor!=null)
                cursor.close();
```

```
        }
    }
    @Override
    public void deleteLinkDevice(int workerId, int deviceId, Response<Boolean> response) {
        try (SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase()) {
            long rowCount = sqLiteDatabase.delete(TABLE_LINK,
                    WORKER_ID_FK + " =? AND " + DEVICE_ID_FK + " =? ",
                    new String[]{String.valueOf(workerId), String.valueOf(deviceId)});
            if (rowCount > 0)
                response.onSuccess(true);
            else
                response.onFailure("Delete error");
        } catch (Exception e) {
            response.onFailure(e.getMessage());
        }
    }
}
```

LinkListAdapter.java

```
package it_school.sumdu.edu.linkcontrol;


import android.annotation.SuppressLint;
import android.content.Context;
import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Device;
import java.util.List;

public class LinkListAdapter extends RecyclerView.Adapter<LinkViewHolder> {

    private final Context context;
    private final int workerId;
    private final List<Device> deviceList;
    private final LinkListener listener;
    public LinkListAdapter(Context context, int workerId, List<Device> deviceList, LinkListener listener) {
        this.context = context;
        this.workerId = workerId;
        this.deviceList = deviceList;
        this.listener = listener;
    }
    @NonNull
    @Override
    public LinkViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.item_linker_device, parent, false);
        return new LinkViewHolder(view);
    }
    @Override
    public void onBindViewHolder(@NonNull LinkViewHolder holder, int position) {
        final int itemPos = position;
        final Device device = deviceList.get(position);
        holder.deviceNameTextView.setText(device.getName());
```

```
    holder.deviceTypeTextView.setText(device.getType());
    holder.deviceModelTextView.setText(device.getModel());
    holder.deleteIcon.setOnClickListener(v -> removeLinkDevice(device.getId(), itemPos));
}
@Override
public int getItemCount() {
    return deviceList.size();
}
private void removeLinkDevice(int deviceId, final int itemPosition) {
    ContactRequest.LinkRequest request = new LinkImplementation();
    request.deleteLinkDevice(workerId, deviceId, new Response<Boolean>() {
        @SuppressLint("NotifyDataSetChanged")
        @Override
        public void onSuccess(Boolean data) {
            if(data){
                deviceList.remove(itemPosition);
                notifyDataSetChanged();
                listener.onLinkUpdated(true);
            } else
                Toast.makeText(context, "Failed to remove device", Toast.LENGTH_LONG).show();
        }
        @Override
        public void onFailure(String message) {
            Toast.makeText(context, message, Toast.LENGTH_LONG).show();
        }
    });
}
```

```
}
```

LinkListener.java

```
package it_school.sumdu.edu.linkcontrol;


public interface LinkListener {
    void onLinkUpdated(boolean isUpdated);
}
```

LinkViewHolder.java

```
package it_school.sumdu.edu.linkcontrol;


import androidx.recyclerview.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import it_school.sumdu.edu.linkcontrol.R;


class LinkViewHolder extends RecyclerView.ViewHolder {
    TextView deviceNameTextView;
    TextView deviceTypeTextView;
    TextView deviceModelTextView;
    ImageView deleteIcon;
    public LinkViewHolder(View itemView) {
        super(itemView);
        deviceNameTextView = itemView.findViewById(R.id.deviceNameTextView);
        deviceTypeTextView = itemView.findViewById(R.id.deviceTypeTextView);
        deviceModelTextView = itemView.findViewById(R.id.deviceModelTextView);
        deleteIcon = itemView.findViewById(R.id.deleteIcon);
```

```
    }
}
```

Response.java

```
package it_school.sumdu.edu.linkcontrol;


public interface Response<T> {
    void onSuccess(T data);
    void onFailure(String message);
}
```

SplashActivity.java

```
package it_school.sumdu.edu.linkcontrol;


import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.Window;
import android.widget.ProgressBar;
import androidx.appcompat.app.AppCompatActivity;
import java.util.Timer;
import java.util.TimerTask;


@SuppressLint("CustomSplashScreen")
public class SplashActivity extends AppCompatActivity{
    ProgressBar pb;
    int counter = 0;
    protected void onCreate(Bundle savedInstanceState) {
```

```
        new Handler().postDelayed(() -> {
            Intent i = new Intent(SplashActivity.this, Authorization.class);
            startActivity(i);
            finish();
        }, 3*1000);
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);
        progress();
    }
    public void progress()
    {
        pb = findViewById(R.id.progressBar);
        final Timer t = new Timer();
        TimerTask tt = new TimerTask() {
            @Override
            public void run() {
                counter++;
                pb.setProgress(counter);
                if(counter == 100)
                    t.cancel();
            }
        };
        t.schedule(tt,0,27);
    }
}
```

WorkerCreate.java

```
package it_school.sumdu.edu.linkcontrol;
```

```
import android.app.Dialog;
import android.os.Bundle;
import androidx.fragment.app.DialogFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Worker;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;

import java.util.Objects;



public class WorkerCreate extends DialogFragment {

    private static WorkerListener workerListener;

    private EditText nameEditText;
    private EditText positionEditText;
    private EditText phoneEditText;
    private EditText emailEditText;
    private String nameString = "";
    private String positionString = "";
    private String phoneString = "";
```

```
private String emailString = "";


public WorkerCreate()
{
}
public static WorkerCreate newInstance(String title, WorkerListener listener){
    workerListener = listener;
    WorkerCreate workerCreateDialogFragment = new WorkerCreate();
    Bundle args = new Bundle();
    args.putString("title", title);
    workerCreateDialogFragment.setArguments(args);
   workerCreateDialogFragment.setStyle(DialogFragment.STYLE_NORMAL, R.style.CustomDialog);
    return workerCreateDialogFragment;
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_worker_create, container, false);
    nameEditText = view.findViewById(R.id.workerNameEditText);
    positionEditText = view.findViewById(R.id.positionEditText);
    phoneEditText = view.findViewById(R.id.phoneEditText);
    emailEditText = view.findViewById(R.id.emailEditText);
    Button createButton = view.findViewById(R.id.createButton);
    Button cancelButton = view.findViewById(R.id.cancelButton);
    assert getArguments() != null;
    String title = getArguments().getString(TITLE);
    Objects.requireNonNull(getDialog()).setTitle(title);
```

```
    createButton.setOnClickListener(view1 -> {
        nameString = nameEditText.getText().toString();
        positionString = positionEditText.getText().toString();
        phoneString = phoneEditText.getText().toString();
        emailString = emailEditText.getText().toString();
        final Worker worker = new Worker(-1, nameString, positionString, phoneString, emailString);
        ContactRequest.WorkerRequest workerRequest = new WorkerImplementation();
        workerRequest.createWorker(worker, new Response<Boolean>() {
            @Override
            public void onSuccess(Boolean data) {
                Objects.requireNonNull(getDialog()).dismiss();
                workerListener.onWorkerListUpdate(data);
                Toast.makeText(getContext(), "Worker created successfully", Toast.LENGTH_LONG).show();
            }
            @Override
            public void onFailure(String message) {
                workerListener.onWorkerListUpdate(false);
                Toast.makeText(getContext(), message, Toast.LENGTH_LONG).show();
            }
        });
    });
    cancelButton.setOnClickListener(view12 -> getDialog().dismiss());
    return view;
}
@Override
public void onStart() {
    super.onStart();
```

```
        Dialog dialog = getDialog();
        if (dialog != null) {
            int width = ViewGroup.LayoutParams.MATCH_PARENT;
            int height = ViewGroup.LayoutParams.WRAP_CONTENT;
            dialog.getWindow().setLayout(width, height);
        }
    }
}
```

WorkerImplementation.java

```
package it_school.sumdu.edu.linkcontrol;

import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import it_school.sumdu.edu.linkcontrol.model.Worker;
import java.util.ArrayList;
import java.util.List;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;

public class WorkerImplementation implements ContactRequest.WorkerRequest {
    private final DatabaseHelper databaseHelper = DatabaseHelper.getInstance();
    @Override
    public void createWorker(Worker worker, Response<Boolean> response) {
        try (SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase()) {
            ContentValues contentValues = getContentValuesForWorker(worker);
```

```
        long id = sqLiteDatabase.insertOrThrow(TABLE_WORKERS, null, contentValues);
        if (id > 0) {
            response.onSuccess(true);
            worker.setId((int) id);
        } else
            response.onFailure("Failed to create student. Unknown Reason!");
    } catch (SQLiteException e) {
        response.onFailure(e.getMessage());
    }
}
@Override
public void readWorker(int workerId, Response<Worker> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();
    Cursor cursor = null;
    try {
        cursor = sqLiteDatabase.query(TABLE_WORKERS, null,
                WORKER_ID + " =? ", new String[]{String.valueOf(workerId)},
                null, null, null);
        if(cursor!=null && cursor.moveToFirst()) {
            Worker worker = getWorkerFromCursor(cursor);
            response.onSuccess(worker);
        }
        else
            response.onFailure("Student not found with this ID in database");
    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
```

```
        sqLiteDatabase.close();
        if(cursor!=null)
            cursor.close();
    }
}
public void readAllWorker(Response<List<Worker>> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getReadableDatabase();
    List<Worker> workerList = new ArrayList<>();
    Cursor cursor = null;
    try {
        cursor = sqLiteDatabase.query(TABLE_WORKERS, null, null, null,
                null, null, null);
        if(cursor!=null && cursor.moveToFirst()){
            do {
                Worker worker = getWorkerFromCursor(cursor);
                workerList.add(worker);
            } while (cursor.moveToNext());
            response.onSuccess(workerList);
        } else
            response.onFailure("Worker list is empty");
    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
        if(cursor!=null)
            cursor.close();
    }
```

```
}
@Override
public void updateWorker(Worker worker, Response<Boolean> response) {
    SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase();
    ContentValues contentValues = getContentValuesForWorker(worker);
    try {
        long rowCount = sqLiteDatabase.update(TABLE_WORKERS, contentValues,
                WORKER_ID + " =? ", new String[]{String.valueOf(worker.getId())});
        if(rowCount>0)
            response.onSuccess(true);
        else
            response.onFailure("Update worker error");
    } catch (Exception e){
        response.onFailure(e.getMessage());
    } finally {
        sqLiteDatabase.close();
    }
}
@Override
public void deleteWorker(int workerId, Response<Boolean> response) {
    try (SQLiteDatabase sqLiteDatabase = databaseHelper.getWritableDatabase()) {
        long rowCount = sqLiteDatabase.delete(TABLE_WORKERS, WORKER_ID + " =? ",
                new String[]{String.valueOf(workerId)});
        if (rowCount > 0)
            response.onSuccess(true);
        else
            response.onFailure("Delete worker error");
```

```
        } catch (Exception e) {
            response.onFailure(e.getMessage());
        }
    }
    private ContentValues getContentValuesForWorker(Worker worker){
        ContentValues contentValues = new ContentValues();
        contentValues.put(WORKER_NAME, worker.getName());
        contentValues.put(WORKER_POSITION, worker.getPosition());
        contentValues.put(WORKER_PHONE, worker.getPhone());
        contentValues.put(WORKER_EMAIL, worker.getEmail());
        return contentValues;
    }
    private Worker getWorkerFromCursor(Cursor cursor){
        @SuppressLint("Range") int id = cursor.getInt(cursor.getColumnIndex(WORKER_ID));
        @SuppressLint("Range") String name = cursor.getString(cursor.getColumnIndex(WORKER_NAME));
        @SuppressLint("Range") String position = cursor.getString(cursor.getColumnIndex(WORKER_POSITION));
        @SuppressLint("Range") String phone = cursor.getString(cursor.getColumnIndex(WORKER_PHONE));
        @SuppressLint("Range") String email = cursor.getString(cursor.getColumnIndex(WORKER_EMAIL));
        return new Worker(id, name, position, phone, email);
    }
}
```

WorkerListActivity.java

```
package it_school.sumdu.edu.linkcontrol;


import android.annotation.SuppressLint;
import android.content.Intent;
import android.os.Bundle;
```

```
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;
import androidx.appcompat.widget.Toolbar;
import android.view.View;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import it_school.sumdu.edu.linkcontrol.model.Worker;
import it_school.sumdu.edu.linkcontrol.model.Count;
import it_school.sumdu.edu.linkcontrol.stat.Constant;
import java.util.ArrayList;
import java.util.List;

public class WorkerListActivity extends AppCompatActivity implements WorkerListener {
    private RecyclerView recyclerView;
    private TextView noDataFoundTextView;
    private FloatingActionButton fab;
    private TextView workerCountTextView;
    private TextView deviceCountTextView;
    private TextView linkCountTextView;
    private final List<Worker> workerList = new ArrayList<>();
    private WorkerListAdapter adapter;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_worker_list);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    initialization();
    adapter = new WorkerListAdapter(this, workerList, this);
    recyclerView.setLayoutManager(new LinearLayoutManager(this, LinearLayoutManager.VERTICAL, false));
    recyclerView.setAdapter(adapter);
    showWorkerList();
    fab.setOnClickListener(view -> {
        WorkerCreate workerCreate = WorkerCreate.newInstance("Create Worker", WorkerListActivity.this);
        workerCreate.show(getSupportFragmentManager(), Constant.CREATE_WORKER);
    });
}
@Override
protected void onResume() {
    super.onResume();
    showTableRowCount();
}
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
```

```
    if(id == R.id.action_show_device){
        startActivity(new Intent(this, DeviceListActivity.class));
    }
    return super.onOptionsItemSelected(item);
}
@Override
public void onWorkerListUpdate(boolean isUpdated) {
    if(isUpdated) {
        showWorkerList();
        showTableRowCount();
    }
}
private void showWorkerList() {
    ContactRequest.WorkerRequest query = new WorkerImplementation();
    query.readAllWorker(new Response<List<Worker>>() {
        @SuppressLint("NotifyDataSetChanged")
        @Override
        public void onSuccess(List<Worker> data) {
            recyclerView.setVisibility(View.VISIBLE);
            noDataFoundTextView.setVisibility(View.GONE);
            workerList.clear();
            workerList.addAll(data);
            adapter.notifyDataSetChanged();
        }
        @Override
        public void onFailure(String message) {
            recyclerView.setVisibility(View.GONE);
```

```
            noDataFoundTextView.setVisibility(View.VISIBLE);
        }
    });
}
private void showTableRowCount() {
    ContactRequest.CountRequest query = new CountImplementation();
    query.getCount(new Response<Count>() {
        @Override
        public void onSuccess(Count data) {
            workerCountTextView.setText(getString(R.string.worker_count, data.getWorkerRow()));
            deviceCountTextView.setText(getString(R.string.device_count, data.getDeviceRow()));
            linkCountTextView.setText(getString(R.string.link_count, data.getLinkRow()));
        }
        @Override
        public void onFailure(String message) {
            workerCountTextView.setText(getString(R.string.count_error));
            deviceCountTextView.setText(message);
            linkCountTextView.setText("");
        }
    });
}
private void initialization(){
    recyclerView = findViewById(R.id.recyclerView);
    noDataFoundTextView = findViewById(R.id.noDataFoundTextView);
    fab = findViewById(R.id.fab);
    workerCountTextView = findViewById(R.id.workerCount);
    deviceCountTextView = findViewById(R.id.devicesCount);
```

```
        linkCountTextView = findViewById(R.id.linkCount);
    }
}
```

WorkerListAdapter.java

```
package it_school.sumdu.edu.linkcontrol;


import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.recyclerview.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Worker;
import java.util.List;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.*;


public class WorkerListAdapter extends RecyclerView.Adapter<WorkerViewHolder> {
    private final Context context;
    private final List<Worker> workerList;
    private final WorkerListener listener;
   WorkerListAdapter(Context context, List<Worker> workerList, WorkerListener listener) {
        this.context = context;
        this.workerList = workerList;
```

```
        this.listener = listener;
    }
    @NonNull
    @Override
    public WorkerViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(context).inflate(R.layout.item_worker_card_view, parent, false);
        return new WorkerViewHolder(view);
    }
    @Override
    public void onBindViewHolder(@NonNull WorkerViewHolder holder, int position) {
        final Worker worker = workerList.get(position);
        holder.nameTextView.setText(worker.getName());
        holder.positionTextView.setText(worker.getPosition());
        holder.emailTextView.setText(worker.getEmail());
        holder.phoneTextView.setText(worker.getPhone());
        holder.editImageView.setOnClickListener(v -> {
            WorkerUpdate dialogFragment = WorkerUpdate.newInstance(worker, "Update Worker", inUpdated ->
listener.onWorkerListUpdate(inUpdated));
            dialogFragment.show(((WorkerListActivity) context).getSupportFragmentManager(), UPDATE_WORKER);
        });
        holder.deleteImageView.setOnClickListener(v -> showConfirmationDialog(worker.getId()));
        holder.itemView.setOnClickListener(v -> {
            Intent intent = new Intent(context, LinkActivity.class);
            intent.putExtra(WORKER_ID, worker.getId());
            context.startActivity(intent);
        });
    }
    @Override
```

```
public int getItemCount() {
    return workerList.size();
}
private void showConfirmationDialog(final int workerId) {
    AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(context);
    alertDialogBuilder.setMessage("Are you sure, You wanted to delete this worker?");
    alertDialogBuilder.setPositiveButton("Yes",
            (arg0, arg1) -> {
                ContactRequest.WorkerRequest query = new WorkerImplementation();
                query.deleteWorker(workerId, new Response<Boolean>() {
                    @Override
                    public void onSuccess(Boolean data) {
                        if(data) {
                            Toast.makeText(context, "Worker deleted successfully", Toast.LENGTH_SHORT).show();
                            listener.onWorkerListUpdate(true);
                        }
                    }
                    @Override
                    public void onFailure(String message) {
                        Toast.makeText(context, message, Toast.LENGTH_LONG).show();
                    }
                });
            });
    alertDialogBuilder.setNegativeButton("No", (dialog, which) -> dialog.dismiss());
    AlertDialog alertDialog = alertDialogBuilder.create();
    alertDialog.show();
}
```

```
}
```

WorkerListener.java

```
package it_school.sumdu.edu.linkcontrol;


public interface WorkerListener {
    void onWorkerListUpdate(boolean isUpdated);
}
```

WorkerUpdate.java

```
package it_school.sumdu.edu.linkcontrol;


import android.app.Dialog;
import android.os.Bundle;
import androidx.fragment.app.DialogFragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import it_school.sumdu.edu.linkcontrol.model.Worker;
import static it_school.sumdu.edu.linkcontrol.stat.Constant.TITLE;
import java.util.Objects;


public class WorkerUpdate extends DialogFragment {


    private static WorkerListener workerListener;
    private EditText nameEditText;
```

```
private EditText positionEditText;
private EditText phoneEditText;
private EditText emailEditText;
private String nameString = "";
private String positionString = "";
private String phoneString = "";
private String emailString = "";
private static Worker worker;
public WorkerUpdate() {
}
public static WorkerUpdate newInstance(Worker wrk, String title, WorkerListener listener){
    worker = wrk;
    workerListener = listener;
    WorkerUpdate workerUpdate = new WorkerUpdate();
    Bundle args = new Bundle();
    args.putString("title", title);
    workerUpdate.setArguments(args);
    workerUpdate.setStyle(DialogFragment.STYLE_NORMAL, R.style.CustomDialog);
    return workerUpdate;
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_worker_update, container, false);
    assert getArguments() != null;
    String title = getArguments().getString(TITLE);
    Objects.requireNonNull(getDialog()).setTitle(title);
```

```
nameEditText = view.findViewById(R.id.workerNameEditText);
positionEditText = view.findViewById(R.id.positionEditText);
phoneEditText = view.findViewById(R.id.phoneEditText);
emailEditText = view.findViewById(R.id.emailEditText);
Button updateButton = view.findViewById(R.id.updateButton);
Button cancelButton = view.findViewById(R.id.cancelButton);
nameEditText.setText(worker.getName());
positionEditText.setText(worker.getPosition());
phoneEditText.setText(worker.getPhone());
emailEditText.setText(worker.getEmail());
updateButton.setOnClickListener(view1 -> {
    nameString = nameEditText.getText().toString();
    positionString = positionEditText.getText().toString();
    phoneString = phoneEditText.getText().toString();
    emailString = emailEditText.getText().toString();
    worker.setName(nameString);
    worker.setPosition(positionString);
    worker.setPhone(phoneString);
    worker.setEmail(emailString);
    ContactRequest.WorkerRequest workerRequest = new WorkerImplementation();
    workerRequest.updateWorker(worker, new Response<Boolean>() {
        @Override
        public void onSuccess(Boolean data) {
            Objects.requireNonNull(getDialog()).dismiss();
            workerListener.onWorkerListUpdate(data);
            Toast.makeText(getContext(), "Worker updated successfully", Toast.LENGTH_LONG).show();
        }
```

```
            @Override
            public void onFailure(String message) {
                workerListener.onWorkerListUpdate(false);
                Toast.makeText(getContext(), message, Toast.LENGTH_LONG).show();
            }
        });
    });
    cancelButton.setOnClickListener(view12 -> getDialog().dismiss());
    return view;
  }
  @Override
  public void onStart() {
    super.onStart();
    Dialog dialog = getDialog();
    if (dialog != null) {
      int width = ViewGroup.LayoutParams.MATCH_PARENT;
      int height = ViewGroup.LayoutParams.WRAP_CONTENT;
      dialog.getWindow().setLayout(width, height);
    }
  }
}
```

WorkerViewHolder.java

```
package it_school.sumdu.edu.linkcontrol;

import androidx.recyclerview.widget.RecyclerView;
import android.view.View;
import android.widget.ImageView;
```

```
import android.widget.TextView;

import it_school.sumdu.edu.linkcontrol.R;


class WorkerViewHolder extends RecyclerView.ViewHolder {
    TextView nameTextView;
    TextView positionTextView;
    TextView emailTextView;
    TextView phoneTextView;
    ImageView editImageView;
    ImageView deleteImageView;
    WorkerViewHolder(View itemView) {
        super(itemView);
        nameTextView = itemView.findViewById(R.id.nameTextView);
        positionTextView = itemView.findViewById(R.id.workerPositionTextView);
        emailTextView = itemView.findViewById(R.id.emailTextView);
        phoneTextView = itemView.findViewById(R.id.phoneTextView);
        editImageView = itemView.findViewById(R.id.editImageView);
        deleteImageView = itemView.findViewById(R.id.deleteImageView);
    }
}
}
```

activity_auth.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:orientation="vertical"
    android:layout_height="match_parent"
```

```
android:background="@color/white">

<ImageView
    android:id="@+id/imageView"
    android:layout_width="200dp"
    android:layout_height="200dp"
    android:layout_gravity="center_horizontal|center_vertical"
    android:layout_marginTop="50dp"
    android:contentDescription="@string/app_logo"
    android:paddingBottom="50dp"
    android:src="@drawable/app_logo"
    app:layout_constraintTop_toTopOf="parent"
    tools:layout_editor_absoluteX="105dp" />
<TextView
    android:layout_width="200dp"
    android:layout_height="150dp"
    android:layout_gravity="center_horizontal|center_vertical"
    android:text="@string/auth_instruction"
    android:textStyle="italic"
    android:textAlignment="center"
    android:textSize="14sp"
    />
<Button
    android:id="@+id/btnAuthorization"
    android:layout_width="170dp"
    android:layout_height="50dp"
    android:layout_gravity="center_horizontal|center_vertical"
    android:layout_marginTop="20dp"
    android:onClick="Sign"
    android:text="@string/authorization"
    app:backgroundTint="@color/grey"
    android:textColor="@color/black"
```

```
        tools:ignore="UsingOnClickInXml" />



</LinearLayout>
```

activity_device_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/activity_background"
    tools:context=".DeviceListActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay"
            app:title="@string/device_title" />

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_devices" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
```

```
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        app:srcCompat="@drawable/ic_add_device" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

activity_link.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DeviceAddActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toTopOf="@id/closeButton"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:listitem="@layout/item_link_card_view">

    </androidx.recyclerview.widget.RecyclerView>

    <TextView
        android:id="@+id/noDataFoundTextView"
        android:layout_width="wrap_content"
```

```
        android:gravity="center"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toTopOf="parent"
        android:padding="8dp"
        app:layout_constraintBottom_toTopOf="@id/closeButton"
        android:text="@string/device_DB_empty"
        tools:ignore="MissingConstraints" />

    <Button
        android:id="@+id/closeButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="closeActivity"
        android:text="@string/close"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_worker_info.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="8dp"
    tools:context=".LinkActivity">

    <include
        android:id="@+id/studentInfoLayout"
        layout="@layout/layout_worker_info"
        android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content" />

<TextView
    android:id="@+id/titleLabel"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="16dp"
    android:background="@color/colorPrimary"
    android:gravity="center_vertical"
    android:padding="8dp"
    android:text="@string/worker_info"
    android:textColor="@android:color/white"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@id/studentInfoLayout" />

<ImageView
    android:id="@+id/action_add_device"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="?android:attr/selectableItemBackground"
    android:padding="4dp"
    app:layout_constraintBottom_toBottomOf="@id/titleLabel"
    app:layout_constraintEnd_toEndOf="@id/titleLabel"
    app:layout_constraintTop_toTopOf="@id/titleLabel"
    app:srcCompat="@drawable/ic_add_device" />

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginTop="8dp"
```

```
        app:layout_constraintBottom_toTopOf="@id/rowCountLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/titleLabel"
        tools:listitem="@layout/item_linker_device">

    </androidx.recyclerview.widget.RecyclerView>

    <TextView
        android:id="@+id/noDataFoundTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        tools:text="@string/empty_list"
        app:layout_constraintBottom_toTopOf="@id/rowCountLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/titleLabel" />

    <include
        android:id="@+id/rowCountLayout"
        layout="@layout/layout_stats"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

activity_worker_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.coordinatorlayout.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/activity_background"
    tools:context="it_school.sumdu.edu.linkcontrol.WorkerListActivity">

    <com.google.android.material.appbar.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/AppTheme.AppBarOverlay">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:popupTheme="@style/AppTheme.PopupOverlay"
            app:title="@string/app_name"/>

    </com.google.android.material.appbar.AppBarLayout>

    <include layout="@layout/content_main" />

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="16dp"
        app:srcCompat="@drawable/ic_worker_add" />

</androidx.coordinatorlayout.widget.CoordinatorLayout>
```

authorization.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.appcompat.widget.LinearLayoutCompat xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <EditText
        android:id="@+id/mailFieldAuth"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:autofillHints=""
        android:ems="10"
        android:inputType="textEmailAddress"
        android:text="@string/enter_your_mail"
        android:layout_marginStart="50dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:ignore="LabelFor" />
    <EditText
        android:id="@+id/passwordFieldAuth"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:ems="10"
        android:inputType="textPassword"
        android:text="@string/enter_your_password"
        android:layout_marginTop="20dp"
        android:layout_marginStart="50dp"
        app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/mailFieldReg"
        app:layout_constraintVertical_bias="0.204"
        android:autofillHints=""
        tools:ignore="LabelFor" />

</androidx.appcompat.widget.LinearLayoutCompat>
```

content_devices.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context=".WorkerListActivity"
    tools:showIn="@layout/activity_device_list">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recyclerView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:padding="8dp"
        app:layout_constraintBottom_toTopOf="@id/rowCountLayout"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:listitem="@layout/item_device_card_view">

    </androidx.recyclerview.widget.RecyclerView>
```

```
    <include
        android:id="@+id/rowCountLayout"
        layout="@layout/layout_stats"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <TextView
        android:id="@+id/noDataFoundTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/empty_DB"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:visibility="visible" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

content_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_behavior="@string/appbar_scrolling_view_behavior"
    tools:context="it_school.sumdu.edu.linkcontrol.WorkerListActivity"
```

```
tools:showIn="@layout/activity_worker_list">

<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:padding="8dp"
    app:layout_constraintBottom_toTopOf="@id/rowCountLayout"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    tools:listitem="@layout/item_worker_card_view">

</androidx.recyclerview.widget.RecyclerView>

<include
    android:id="@+id/rowCountLayout"
    layout="@layout/layout_stats"
    android:layout_width="0dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    android:layout_height="wrap_content"
    app:layout_constraintBottom_toBottomOf="parent"/>

<TextView
    android:id="@+id/noDataFoundTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/empty_DB"
    android:visibility="gone"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:visibility="visible" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_device_create.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp">

    <EditText
        android:id="@+id/deviceNameEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/device_name"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/deviceTypeEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/device_type_n"
        android:inputType="text"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/deviceNameEditText" />

    <EditText
```

```
    android:id="@+id/deviceModelEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/device_model_n"
    android:inputType="text"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/deviceTypeEditText" />


  <Button
    android:id="@+id/createButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="@string/add"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toLeftOf="@+id/cancelButton"
    app:layout_constraintTop_toBottomOf="@+id/deviceModelEditText" />

  <Button
    android:id="@+id/cancelButton"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:text="@string/cancel"
    app:layout_constraintLeft_toRightOf="@+id/createButton"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="@id/createButton" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_device_update.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent"
android:padding="10dp">

<EditText
    android:id="@+id/deviceNameEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/device_name"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/deviceTypeEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/device_type"
    android:inputType="text"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/deviceNameEditText" />

<EditText
    android:id="@+id/deviceModelEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/device_model"
    android:inputType="text"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/deviceTypeEditText" />
```

```
    <Button
        android:id="@+id/updateButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="@string/update_button"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/cancelButton"
        app:layout_constraintTop_toBottomOf="@+id/deviceModelEditText" />

    <Button
        android:id="@+id/cancelButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:text="@string/cancel"
        app:layout_constraintLeft_toRightOf="@+id/updateButton"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@id/updateButton" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_worker_create.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:padding="10dp">

    <EditText
        android:id="@+id/workerNameEditText"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
```

```
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:hint="@string/worker_name"
    android:inputType="textPersonName"/>

<EditText
    android:id="@+id/positionEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/workerNameEditText"
    android:hint="@string/position"
    android:inputType="text"/>

<EditText
    android:id="@+id/phoneEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/positionEditText"
    android:hint="@string/phone"
    android:inputType="phone"/>

<EditText
    android:id="@+id/emailEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/phoneEditText"
        android:hint="@string/email"
        android:inputType="textEmailAddress"/>

    <Button
        android:id="@+id/createButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/emailEditText"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/cancelButton"
        android:text="@string/add"/>

    <Button
        android:id="@+id/cancelButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/emailEditText"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/createButton"
        android:text="@string/cancel"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

fragment_worker_update.xml

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:padding="10dp">

    <EditText
```

```
    android:id="@+id/workerNameEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:hint="@string/name"
    android:inputType="textPersonName"/>

<EditText
    android:id="@+id/positionEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/workerNameEditText"
    android:hint="@string/position"
    android:inputType="text"/>

<EditText
    android:id="@+id/phoneEditText"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/positionEditText"
    android:hint="@string/phone"
    android:inputType="phone"/>

<EditText
    android:id="@+id/emailEditText"
    android:layout_width="0dp"
```

```
        android:layout_height="wrap_content"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/phoneEditText"
        android:hint="@string/email"
        android:inputType="textEmailAddress"/>

    <Button
        android:id="@+id/updateButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/emailEditText"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/cancelButton"
        android:text="@string/update_button"/>

    <Button
        android:id="@+id/cancelButton"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        app:layout_constraintTop_toBottomOf="@+id/emailEditText"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/updateButton"
        android:text="@string/cancel"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

item_device_card_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
```

```
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:foreground="?android:attr/selectableItemBackground"
app:cardElevation="4dp"
android:clickable="true"
android:focusable="true"
app:cardUseCompatPadding="true">

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">

    <TextView
        android:id="@+id/deviceNameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="Data Structures" />

    <TextView
        android:id="@+id/deviceTypeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/deviceNameTextView"
        tools:text="@string/device_type" />

    <TextView
        android:id="@+id/deviceModelTextView"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/deviceTypeTextView"
        tools:text="@string/device_model" />

    <ImageView
        android:id="@+id/deleteIcon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:contentDescription="@string/delete_descr"
        android:foreground="?android:attr/selectableItemBackground"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_delete_black" />

    <ImageView
        android:id="@+id/editIcon"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:contentDescription="@string/update_descr"
        android:foreground="?android:attr/selectableItemBackground"
        app:layout_constraintEnd_toStartOf="@id/deleteIcon"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_edit_black" />

  </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

item_link_card_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:foreground="?android:attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    app:cardUseCompatPadding="true">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp">

        <CheckBox
            android:id="@+id/checkbox"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <TextView
            android:id="@+id/deviceNameTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            app:layout_constraintStart_toEndOf="@id/checkbox"
            android:layout_marginStart="16dp"
            android:textStyle="bold"
            app:layout_constraintTop_toTopOf="parent"
```

```
        tools:text="Data Structures" />

    <TextView
        android:id="@+id/deviceTypeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="@id/deviceNameTextView"
        app:layout_constraintTop_toBottomOf="@id/deviceNameTextView"
        tools:text="@string/device_type" />

    <TextView
        android:id="@+id/deviceModelTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="@id/deviceNameTextView"
        app:layout_constraintTop_toBottomOf="@id/deviceTypeTextView"
        tools:text="@string/device_model" />

  </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

item_linker_device.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:foreground="?android:attr/selectableItemBackground"
  app:cardElevation="4dp"
```

```
android:clickable="true"
android:focusable="true"
app:cardUseCompatPadding="true">

<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp">

    <TextView
        android:id="@+id/deviceNameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="Personal Computer Number 4" />

    <TextView
        android:id="@+id/deviceTypeTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/deviceNameTextView"
        tools:text="@string/device_type" />

    <TextView
        android:id="@+id/deviceModelTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@id/deviceTypeTextView"
```

```
            tools:text="@string/device_model" />

        <ImageView
            android:id="@+id/deleteIcon"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/delete_descr"
            android:foreground="?android:attr/selectableItemBackground"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/ic_delete_black" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

item_worker_card_view.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:foreground="?android:attr/selectableItemBackground"
    app:cardElevation="4dp"
    app:cardUseCompatPadding="true">

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:padding="8dp">
```

```
        <include
            android:id="@+id/workerInfoLayout"
            layout="@layout/layout_worker_info"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />

        <ImageView
            android:id="@+id/editImageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginEnd="8dp"
            android:contentDescription="@string/update_descr"
            android:foreground="?android:attr/selectableItemBackground"
            app:layout_constraintRight_toLeftOf="@+id/deleteImageView"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/ic_edit_black" />

        <ImageView
            android:id="@+id/deleteImageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/delete_descr"
            android:foreground="?android:attr/selectableItemBackground"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:srcCompat="@drawable/ic_delete_black" />

    </androidx.constraintlayout.widget.ConstraintLayout>

</androidx.cardview.widget.CardView>
```

layout_stats.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="#c1fff0"
    android:padding="10dp">

    <TextView
        android:id="@+id/workerCount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@id/devicesCount"
        app:layout_constraintStart_toStartOf="@id/devicesCount"
        tools:text="@string/worker_count" />

    <TextView
        android:id="@+id/devicesCount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@id/linkCount"
        app:layout_constraintStart_toStartOf="@id/linkCount"
        tools:text="@string/device_count" />

    <TextView
        android:id="@+id/linkCount"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        tools:text="@string/link_count" />
```

```
</androidx.constraintlayout.widget.ConstraintLayout>
```

layout_worker_info.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools">

    <TextView
        android:id="@+id/nameTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="John Doe" />

    <TextView
        android:id="@+id/workerPositionTitle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/position"
        android:textStyle="bold"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@id/nameTextView" />

    <TextView
        android:id="@+id/workerPositionTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
    android:layout_marginStart="4dp"
    app:layout_constraintLeft_toRightOf="@id/workerPositionTitle"
    app:layout_constraintTop_toBottomOf="@id/nameTextView"
    tools:text="HR manager" />

<TextView
    android:id="@+id/emailTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/email"
    android:textStyle="bold"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@id/workerPositionTitle" />

<TextView
    android:id="@+id/emailTextView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    app:layout_constraintLeft_toRightOf="@id/emailTitle"
    app:layout_constraintTop_toBottomOf="@id/workerPositionTitle"
    tools:text="john@gmail.com" />

<TextView
    android:id="@+id/phoneTitle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/phone"
    android:textStyle="bold"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintTop_toBottomOf="@id/emailTitle" />
```

```
    <TextView
        android:id="@+id/phoneTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="4dp"
        app:layout_constraintLeft_toRightOf="@id/phoneTitle"
        app:layout_constraintTop_toBottomOf="@id/emailTitle"
        tools:text="+380503459382" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

splash.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    android:orientation="vertical"
    android:weightSum="1">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_gravity="center_horizontal|center_vertical"
        android:contentDescription="@string/app_logo"
        android:paddingBottom="50dp"
        android:src="@drawable/app_logo"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

```
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal|bottom"
    android:text="@string/app_name"
    android:textColor="@color/black"
    android:textSize="40sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toTopOf="@+id/progressBar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/imageView" />

<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_gravity="center_horizontal|bottom"
    android:text="@string/app_descr"
    android:textColor="@color/black"
    android:textSize="14sp"
    android:textStyle="italic"
    app:layout_constraintBottom_toTopOf="@+id/progressBar"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<ProgressBar
    android:id="@+id/progressBar"
```

```
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="match_parent"
        android:layout_height="59dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```