МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ЦЕНТР ЗАОЧНОЇ, ДИСТАНЦІЙНОЇ ТА ВЕЧІРНЬОЇ ФОРМ НАВЧАННЯ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Проектування і розробка веб-додатку для клієнтів підприємства «Сумські телекомсистеми»»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Кузіков Б.О.**

Студента групи Індн-61С — **Пархомчук А.К.**

СУМИ 2020

ЗМІСТ

ЗМІСТ 2
ВСТУП 3
1 ПЕРЕГЛЯД ІСНУЮЧИХ РІШЕНЬ 4
1.1 Поточне рішення підприємства 4
1.2 Приклади сторінок «Особистий кабінет» 7
1.3 Перегляд СУБД 10
1.3.1 Приклади СУБД 11
1.4 Огляд мов програмування 13
2 ВИБІР СПОСОБУ РІШЕННЯ 15
2.1 Проектування інформаційної системи 15
2.1.1 Побудова діаграм потоків даних 18
2.1.2 Логічна структура бази даних 21
2.2 Фізична реалізація бази даних 26
2.2.1 Створення ключових сутностей бази даних 27
2.3 Проектування веб-додатку 30
2.3.1 Опис архітектури веб-додатку 30
2.3.2 Створення UML-діаграм 32
3 ПРОГРАМНА РЕАЛІЗАЦІЯ 35
3.1 Створення й настройка додатку 35
3.1.1 Вибір середовища розробки для створення додатку 35
3.1.2 Налаштування системи збірки Maven 37
3.1.3 Використання системи контролю версій 40
3.1.4 Налаштування веб-сервера і розгортання додатку 44
3.2 Опис графічного інтерфейсу 46
ВИСНОВКИ 52
СПИСОК ЛІТЕРАТУРИ 53
ДОДАТОК 1 54
ДОДАТОК 2 55
ДОДАТОК 3 56
ДОДАТОК 4 59

ВСТУП

У наш час телекомунікаційні компанії постійно вдосконалюють свої інформаційні структури і додатки для розширення функціоналу та можливостей взаємодії з користувачами. Це дозволяє їм підвищити конкурентоспроможність на ринку, а також зробити більш зручний доступ до призначених для користувача послуг клієнтами підприємства. Актуальність нашої роботи полягає в створенні інформаційної системи, яка дасть підприємству точну, своєчасну, актуальну і повну інформацію, необхідну для прийняття важливих рішень всередині компанії.

Метою дипломної роботи було освоєння під безпосереднім керівництвом викладача практичними прийомами і навичками проектування інформаційної системи, застосування отриманих теоретичних знань в області «Інформатика».

Нашим завданням було удосконалити інформаційну систему підприємства, створивши веб-додаток для клієнтів, за допомогою якого вони зможуть здійснювати взаємодію з системою підприємства: вносити зміни в надані телекомунікаційні послуги, переглядати історію платежів, здійснювати оплату рахунків, отримувати інформацію про компанію та про надані нею послуги .

Прикладом застосування отриманих результатів є реалізація нової програми для розширення функціональності ТОВ «Сумські телекомсистеми», де пріоритетним напрямком розвитку в сфері надання послуг доступу до мережі Інтернет є побудова високошвидкісних каналів зв'язку.

Практична значимість даної роботи полягає в тому, що результати роботи можуть бути корисними при використанні компанією для удосконалення управління інформаційною системою підприємства.

1 ПЕРЕГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Поточне рішення підприємства

На даний момент на підприємстві ООО «Сумські телекомсистеми» існує декілька видів взаємодії клієнтів з оператором зв’язку:

– веб-сайт http://tks.sumy.ua;

– телефони технічної підтримки;

– електронна пошта;

– сервісний центр.

Веб-сайт http://tks.sumy.ua є сайтом-візиткою для підприємства (рисунок 1.1).

Рисунок 1.1 Головна сторінка сайту підприємства.

Основні завдання сайту:

- залучення потенціальних клієнтів з мережи Інтернет;
- надання інформації про услуги, тарифи й акції компанії;
- надання контактної інформації для безпосереднього зв’язку з компанією.

На сайті також знаходяться посилання на нормативні документи підприємства й необхідна інформація для приватних клієнтів, для бізнес-клієнтів і для будівельних організацій.

Основні переваги сайту-візитки наступні:

- відносно низька вартість створення и розробки сайту;
- сайт не потребує особливих витрат на обслуговування й постійне оновлення контенту;
- досить простій сайт, надає тільки найважливішу інформацію;
- можливість модернізувати сайт у майбутньому.

Основні недоліки сайту-візитки:

- неможливо автоматизувати процеси, тому потрібен додатковий штат співробітників для роботи з клієнтами.

Таким чином сайт-візитка є простим й зручним рішенням для представлення компанії в мережі Інтернет, але його функціональність досить сильно обмежена.

Телефони технічної підтримки існують для надання різноманітної інформації про підприємство та його послуги. По-перше, користувачі можуть підключати, модифікувати або відключати послуги. По-друге, за допомогою телефонного зв'язку можна вирішити технічні проблеми або проблеми з системою оплати рахунків.

Один з телефонних номерів, вказаних на сайті, є багатоканальним. Багатоканальний номер - це номер, що дозволяє приймати декілька дзвінків одночасно. Дзвінки, що надходять на цей номер, переадресовуються на різні телефони. Як правило, створюється список номерів для переадресації, а

також пріоритетність з'єднання з тим або іншим номером. У будь-якому випадку, на вхідний дзвінок відповідає один з вільних на даний момент операторів.

Компанія опублікувала даний телефонний номер у відкритому доступі, а на дзвінки відповідають спеціально навчені оператори call-центрів. Завдяки багатоканальному номеру клієнтам компанії не доводиться подовгу чекати з'єднання або чути сигнал «зайнято». Кількість одночасних з'єднань може бути встановлено в залежності від потреб фірми і поточного навантаження.

Одним з альтернативних шляхів комунікації користувачів з підприємством є електронна пошта. На сьогоднішній день корпоративна електронна пошта - невід'ємна частина бізнес-культури та бізнес-комунікацій в будь-якій компанії. Вона виконує функцію засобу зв'язку, реклами, просування товару чи послуги. Однак, створення і підтримка сервера корпоративної електронної пошти досить складний процес для системного адміністратора компанії, адже необхідно детально продумати й реалізувати апаратне й програмне забезпечення, налаштувати антивірусний і антиспам-захист майбутнього поштового сервера, а також не забути про зручність користувачів електронної пошти.

По електронній пошті користувачі можуть надсилати свої питання або запити на зміну персональних даних або послуг. На даний момент існує 2 електронні адреси підприємства за якими можна звернутися:

- загальні питання: info@tks.sumy.ua;
- служба технічної підтримки: help@tks.sumy.ua.

Також у клієнтів є можливість звернутися безпосередньо до сервісного центру щоб вирішити всі необхідні питання.

В даному випадку сервісний центр - це організація, що займається наданням послуг з сервісної підтримки, обслуговування телекомунікаційного обладнання та надання консультацій по наданим продуктам.

Адреса та графік роботи сервісного центру можна знайти на сайті компанії або ж уточнити по телефону.

Таким чином, щоб задовольнити потреби клієнтів, підприємство повинно містити вказані вище засоби комунікації. В результаті цього збільшується кількість співробітників, необхідних для виконання цих завдань. Отже, зростають і витрати підприємства.

Ідея даної роботи в тому, щоб автоматизувати частину цих функцій, створивши веб-додаток для клієнтів, за допомогою якого вони зможуть контролювати стан свого рахунку, здійснювати оплату, змінювати особисті дані, а також отримувати необхідну інформацію про послуги, що надаються й управляти ними.

Веб-додаток називається «Особистий кабінет». Особистий кабінет може не тільки допомогти в наданні послуг клієнтам, а й перевести сервіс на більш високий рівень.

Таким чином дане рішення дозволить зняти навантаження з телефонних ліній компанії і зробити комунікацію між користувачами і підприємством більш ефективною та зручною. В результаті успішного впровадження даної програми компанія може зменшити витрати на утримання великого штату обслуговуючого персоналу, так як клієнт самостійно в особистому кабінеті вирішує основну масу завдань.

1.2 Приклади сторінок «Особистий кабінет»

Vodafone

Vodafone - одна з найбільших світових телекомунікаційних компаній, що надає широкий спектр послуг, включаючи мобільний голосовий зв'язок, передачу даних, обмін повідомленнями, фіксований інтернет і кабельне телебачення. Оператор працює в 25 країнах, має партнерські угоди з операторами 44 країн і надає послуги фіксованого широкосмугового доступу в інтернет на 19 ринках.

На головній сторінці веб-сайту можна зручно переглядати стан свого рахунку, витрати, переглянути актуальні тарифи та послуги (рисунок 1.2).

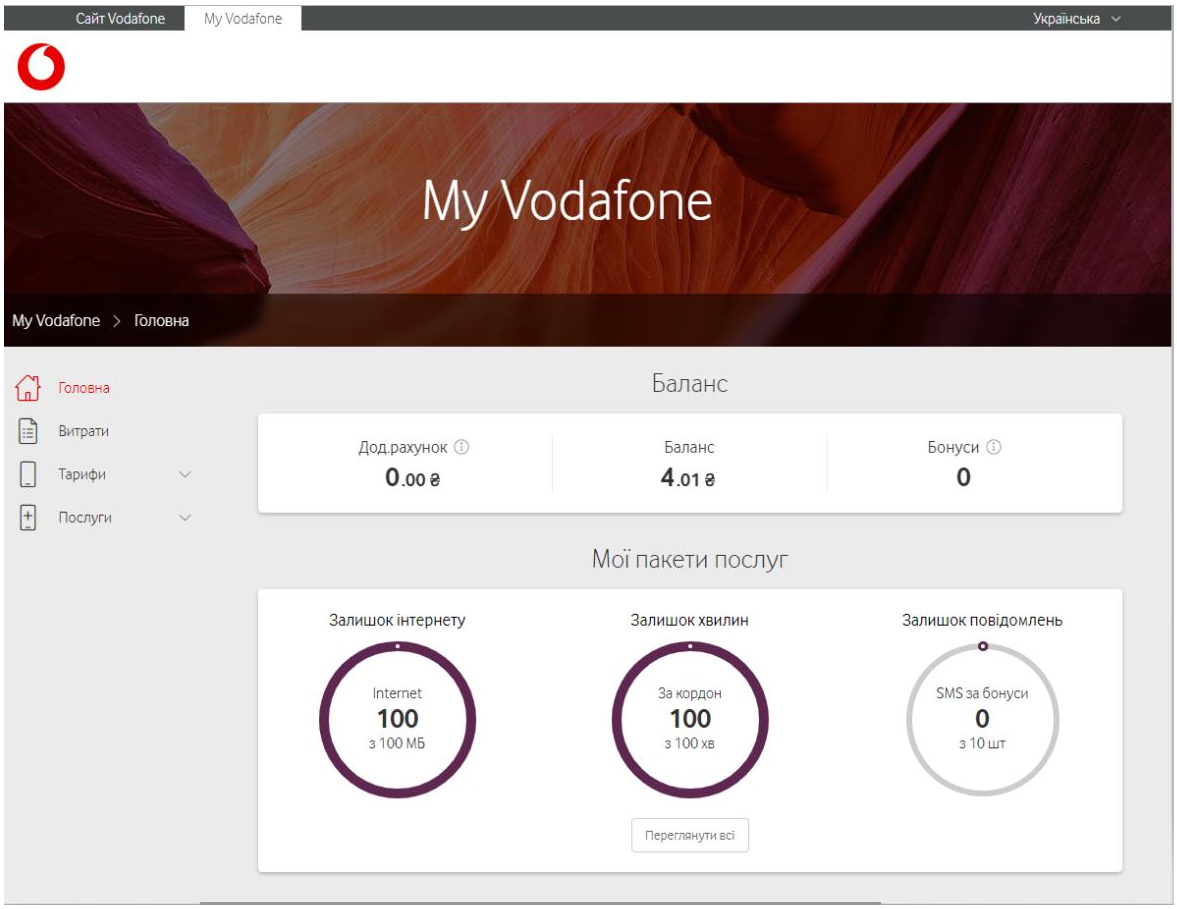


Рисунок 1.2 Домашня сторінка особистого кабінету оператора Vodafone.

Koodo Mobile

Koodo Mobile - оператор мобільного зв'язку в Канаді, утворений компанією Telus в 2008 році. Koodo відрізняється, в основному, тим, що не має довгострокових контрактів, на відміну від свого засновника. Має широке покриття по Канаді, тому користується великою популярністю.

Сайт оператора: https://www.koodomobile.com/

На домашній сторінці можна переглянути поточний баланс, ознайомитися з останніми новинами та акціями, простежити історію своїх дзвінків та їх вартість (рисунок 1.3).

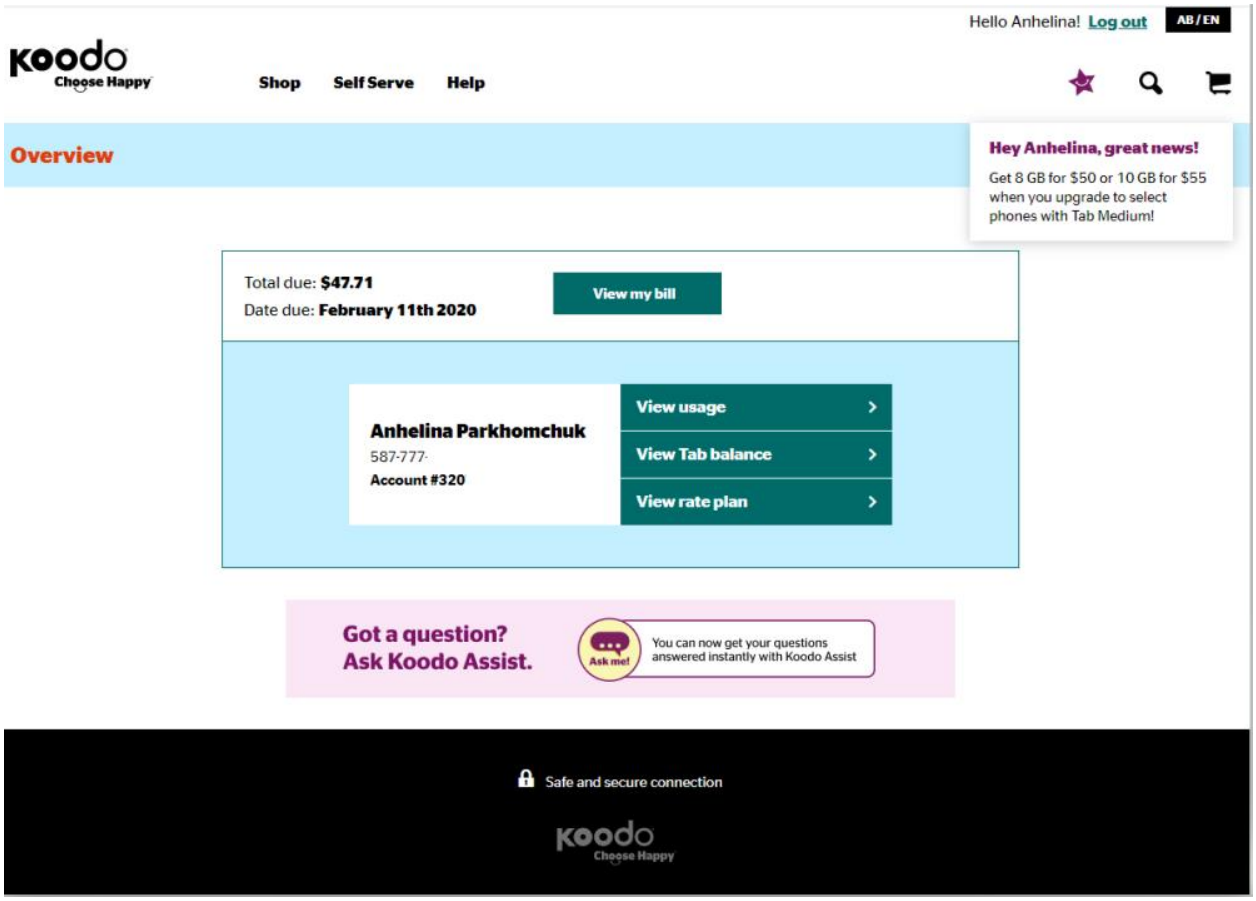


Рисунок 1.3 Сторінка особистого кабінету Koodo Mobile

Shaw Communications Inc

Shaw Communications Inc. - канадська телекомунікаційна компанія, що надає послуги телефонії, Інтернету, телебачення та мобільного зв'язку. Головний офіс розташований в Калгарі, Альберта. Shaw надає послуги головним чином в провінціях Британська Колумбія і Альберта, з меншим переліком послуг у провінціях Саскачеван, Манітоба та Онтаріо. Через свою дочірню компанію Freedom Mobile компанія Shaw надає послуги мобільного зв'язку в міських районах провінцій Британська Колумбія, Альберта і Південний Онтаріо. Головним конкурентом компанії є Telus Corporation.

Сайт компанії: https://www.shaw.ca/store/

На сторінці самообслуговування, аналогічно вищевказаним прикладам, також головним чином прослідковуються операції с балансом та тарифами. Також надана можливість увімкнути послуги щомісячної автоматичної

сплати за користування інтернетом та надсилання електронною поштою рахунку (рисунок 1.4).

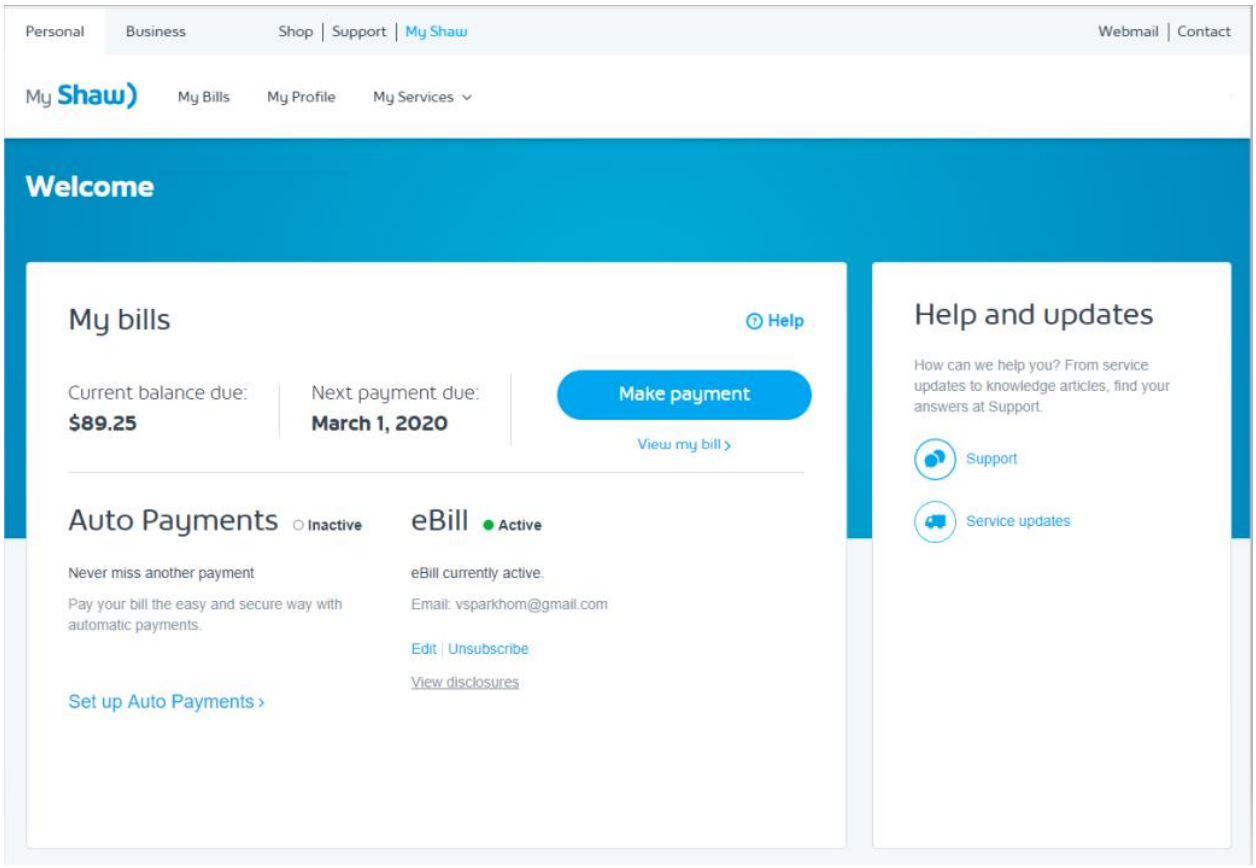


Рисунок 1.4 Сторінка особистого кабінету MyShaw.

1.3 Перегляд СУБД

Система управління базами даних (СУБД) - сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням і використанням баз даних (рисунок 1.5).

Рисунок 1.5 Схема роботи СУБД

Основні функції СУБД:

• управління даними у зовнішній пам'яті (на дисках);

• керування даними в оперативній пам'яті з використанням дискового кешу;

• резервне копіювання і відновлення бази даних після збоїв;

• підтримка мов БД (мова визначення даних, мова маніпулювання даними).

Зазвичай сучасна СУБД містить наступні компоненти:

• ядро, яке відповідає за управління даними у зовнішній і оперативної пам'яті,

• процесор мови бази даних, що забезпечує оптимізацію запитів на вилучення та зміну даних і створення машинно-незалежного виконуваного внутрішнього коду,

• підсистему підтримки часу виконання, яка інтерпретує програми маніпуляції даними, що створюють користувальницький інтерфейс із СУБД,

• сервісні програми (зовнішні утиліти), що забезпечують ряд додаткових можливостей по обслуговуванню інформаційної системи.

1.3.1 Приклади СУБД

Для роботи з базами даних необхідно обрати надійну СУБД. Уміння правильно їх обирати надзвичайно важливе при розробці будь-якого ПО. Переглянемо найосновніші.

MySQL **–** найпопулярніша в світі база даних з відкритим кодом. Завдяки своїй надійності, простоті використання й продуктивності, MySQL використовується для веб-додатків на таких ресурсах, як YouTube, Twitter, Facebook. Особливості MySQL: масштабованість, швидкість, підтримка багатьох операційних систем.

Oracle Database **–** об’єктно-реляційна система управління базами даних компанії Oracle. Дозволяє створювати додатки будь-якого ступеня складності. Завдяки високій ефективності с Oracle Database можуть одночасно працювати безліч користувачів, не знижуючи продуктивності системи. Особливості цієї СУБД: зрозуміла документація, підтримка довгих найменувань, здатність обробляти велику кількість даних.

PostgreSQL **–** вільна об’єктно-реляційна СУБД. Масштабована база даних, що працює на Linux, Windows, OSX і деяких інших системах. У PostgreSQL 10 є такі функції, як логічна реплікація, декларативне розбиття таблиць, більш надійна автентифікація по паролю. Особливості: асинхронна реплікація, відновлення на момент часу, підтримка табличного простору, а також збережених процедур, об’єднань, уявлень і тригерів.

MS SQL Server **–** система керування базами даних, розроблена корпорацією Microsoft. Основна використовувана мова запитів – Transact-SQL, створена спільно Microsoft та Sybase. Графічний інтерфейс і програмне забезпечення базуються на командах. Особливості: висока продуктивність, залежність від платформи, можливість встановити різні версії на одному комп’ютері, генерація скриптів для перенесення даних.

NoSQL рішення:

- **MongoDB –** найпопулярніша NoSQL система управління базами даних. Краще за все підходить для динамічних запитів і визначення індексів. Гнучка структура, яку можливо модифікувати й розширяти. Підтримує Linux, OSX і Windows, але розмір БД лімітовано в 2,5 ГБ. Особливості: висока продуктивність, автоматична фрагментація, робота на декількох серверах, можливість індексувати всі поля в документі.
- **Redis –** СУБД з відкритим вихідним кодом, що використовується для БД, брокерів повідомлень і кешу. Усі операції в Redis атомарні. Написана на мові C і підтримується майже всіма мовами

програмування. Особливості: автоматична обробка відмови, транзакції, сценарії LUA.

- **DB2 –** СУБД, що створена IBM, написана на C, C++ чи Assembly. Вона ідеально підходить для хост-середовищ IBM. Особливостями є покращене вбудоване шифрування і спрощена установка і розгортання.
- **Elasticsearch –** легко масштабована пошукова система корпоративного рівня з відкритим вихідним кодом. Завдяки великому й продуманому API забезпечує надзвичайно швидкий пошук, працює в тому числі з додатками для виявлення даних. Використовується такими компаніями, як Вікіпедія, The Guardian, StackOverflow, GitHub. Особливості: масштабованість до декількох петабайт структурованих і неструктурованих даних, підтримка розрахована на багато користувачів, пошук в режимі реального часу.

1.4 Огляд мов програмування

Розробка веб-додатків - це загальний термін для процесу створення веб-сторінок або сайтів. Ці сторінки можуть містити простий текст і графіку, нагадуючи собою статичний документ. Вони також можуть бути інтерактивними або відображати інформацію, що змінюється. Створювати подібні сторінки трохи складніше, але вони дозволяють розробляти веб-сайти з різноманітним контентом, таким як кошика інтернет-магазинів, динамічна візуалізація і навіть складні соціальні мережі.

На сьогоднішній день існують кілька етапів розробки веб-сайту:

- проектування сайту або веб-додатку (збір і аналіз вимог щодо завдання, розробка технічного завдання, проектування інтерфейсу);
- розробка креативної концепції сайту;
- створення дизайн-концепції сайту;

- створення макетів сторінок;
- створення мультимедіа і FLASH-елементів;
- верстка сторінок і шаблонів;
- програмування (розробка функціональних інструментів)
- оптимізація і розміщення матеріалів сайту;
- тестування і внесення виправлень;
- відкриття проекту;
- обслуговування працюючого сайту або його програмної основи.

Залежно від того на якій стороні проводиться створення веб-додатку, розробку можна розділити на наступні напрямки:

- Front-end розробка
- Back-end розробка

Front-end і back-end - терміни в програмній інженерії, які розрізняють відповідно до принципу поділу відповідальності між зовнішнім поданням і внутрішньою реалізацією відповідно.

В архітектурі програмного забезпечення може бути багато рівнів між апаратною частиною і кінцевим користувачем, кожен з яких також може мати front-end і back-end. Подібне розділення між програмними системами спрощує розробку. Front (також відома як бізнес-процеси, на стороні клієнта) сторона - це будь-який компонент, керований користувачем, а back (сервер) сторона виконується на сервері.

2 ВИБІР СПОСОБУ РІШЕННЯ

2.1 Проектування інформаційної системи

Для забезпечення функціонування веб-додатку «Особистий кабінет» нам необхідно розробити інформаційну систему.

Інформаційна система (ІС) являє собою систему збору, зберігання, обробки, перетворення, передачі та оновлення інформації з використанням комп'ютерної та іншої техніки. Елементами цієї системи є не матеріальні об'єкти, а ті чи інші види даних (інформації), які взаємодіють і перетворюються в процесі її функціонування (рисунок 2.1).

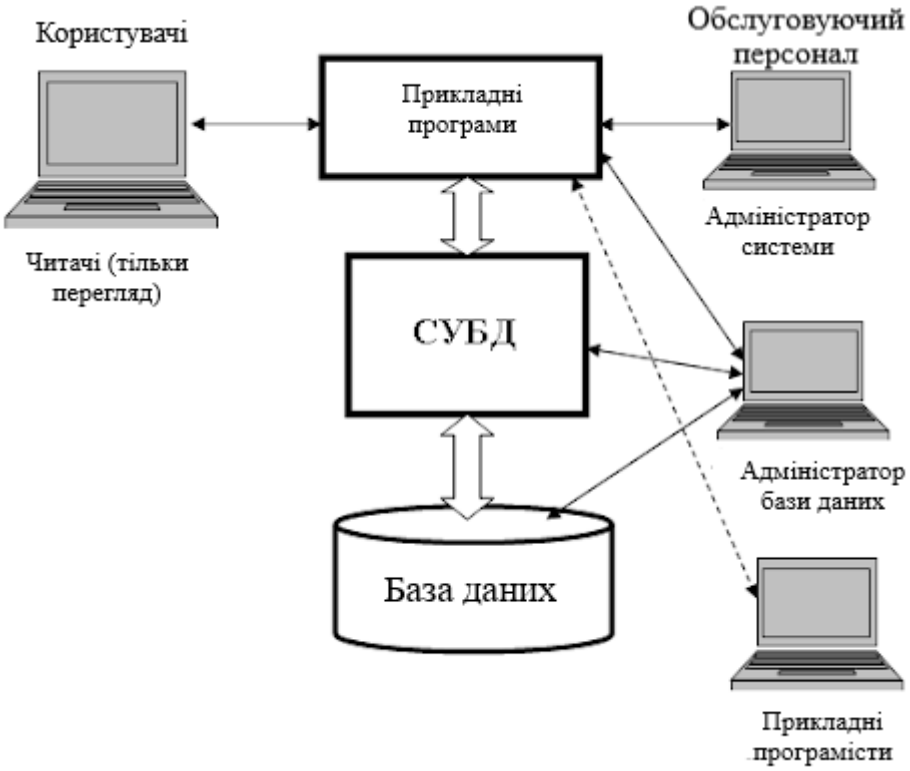


Рисунок 2.1 Компоненти інформаційної системи

Сучасні інформаційні системи вирішують такі основні завдання:

1. Здійснення пошуку, обробки та зберігання інформації, яка накопичується протягом великого періоду часу.
2. Зберігання даних різної структури. Не існує розвиненої ІС, що працює з одним однорідним файлом даних.
3. Аналіз і прогнозування потоків інформації різних видів і типів.

4. Дослідження способів подання і зберігання інформації, розробка спеціальних прийомів стиснення і кодування інформації, анотування об'ємних документів і реферування їх.
5. Побудова процедур і технічних засобів для їх реалізації, за допомогою яких можна автоматизувати процес добування інформації з документів, не призначених для обчислювальних машин, а орієнтованих на сприйняття їх людиною.
6. Створення інформаційно-пошукових систем, здатних сприймати запити до інформаційних сховищ, сформульовані на спеціальних мовах запитів для систем такого типу.
7. Створення мереж зберігання, обробки і передачі інформації, до складу яких входять інформаційні банки даних, термінали, обробні центри та засоби зв'язку.

Створювана нами інформаційна система призначена для забезпечення функцій веб-додатку «Особистий кабінет». Основні користувачі даної системи - це клієнти підприємства ТОВ «Сумські телекомсистеми».

Основний перелік функцій, доступних користувачам особистого кабінету:

- управління телекомунікаційними послугами;
- управління коштами (перегляд історії платежів і оплата рахунків);
- пошук інформації про підприємство та його послуги.

Розглянемо кожну з цих функцій більш докладно.

Управління телекомунікаційними послугами являє собою взаємодію з такими сервісами як: високошвидкісний інтернет, телебачення і телефонний зв'язок. Кожен сервіс має свій набір тарифів, характеристик і параметрів, які можуть бути змінені.

В особистому кабінеті користувач може замовити новий сервіс, відключити або зробити певні модифікації.

У будь-який момент часу в веб-додатку відображається перелік встановлених послуг та список доступних сервісів для підключення. При відключенні існуючої послуги користувачеві потрібно повідомити причину, по якій він відмовляється від даної послуги. При замовленні нової послуги він повинен вказати певні параметри (швидкість, обладнання, кількість каналів, телефонний номер, спосіб оплати і подібні).

Також користувач має можливість в залежності від типу сервісу змінювати деякі параметри. Це можуть бути як загальні параметри (тарифний план), так і більш специфічні для кожного сервісу (швидкість і тип інтернету, кількість телевізійних каналів, телефонний номер і наявність голосової пошти).

Управління коштами включає в себе перегляд історії платежів, здійснення оплати поточного рахунку і настройку автоматичного платежу.

За допомогою історії платежів користувач може простежити динаміку витрат на послуги протягом певного періоду часу, а також обсяг використаних даних (інтернет трафік, кількість телефонних хвилин).

Крім цього, доступна функція оплати рахунку. Це може бути одноразовий платіж або ж настройка автоматичної оплати в разі фіксованого тарифу. Для цього користувач повинен надати дані банківського рахунку або кредитної картки. Функція автоматичної оплати дозволить не пропустити дату платежу та зробить ведення обліку більш простим і зручним для клієнта.

Пошук інформації про підприємство та його послуги складається з декількох складових. Першою складовою є отримання контактних даних підприємства, графік роботи сервісних центрів. Контактні дані можуть в себе включати посилання на інтернет ресурси, номери телефонів, адреси сервісних центрів. У перспективі можлива інтеграція у веб-додаток онлайн підтримки у вигляді чату.

Дана інформація допоможе користувачеві найбільш зручним для нього шляхом вирішити виниклі питання і технічні проблеми.

Другою складовою частиною є дошка оголошень. За допомогою дошки оголошень користувач має можливість бачити у себе на сторінці корисну інформацію про компанію та її послуги. Це можуть бути оголошення про технічні роботи, опис акційних пропозицій та знижок, новини про нововведення підприємства, повідомлення про зміни в тарифних планах. Повідомлення можуть бути як загальними для всіх користувачів, так і індивідуальними, націленими на конкретні інтереси споживача.

Ще однією важливою складовою є наявність розділу найбільш поширених запитань (FAQ, Frequently Asked Questions). За допомогою даного розділу користувач зможе знайти відповіді на питання, що цікавлять його без необхідності залучення персоналу підтримки компанії. У даному розділі можуть бути розміщені відповіді на технічні питання і питання оплати, інструкції з налаштування обладнання та інша корисна інформація.

Всі зазначені вище функції можна зобразити за допомогою діаграм потоків даних. Такі діаграми є основним засобом моделювання функціональних вимог до проектованої системи.

2.1.1 Побудова діаграм потоків даних

Одним з найважливіших етапів проектування ІС є побудова діаграми потоків даних. Діаграми потоків даних (DFD) є основним засобом моделювання функціональних вимог проектованої системи. З їх допомогою ці вимоги розбиваються на функціональні компоненти (процеси) і представляються у вигляді мережі, пов'язаної потоками даних. Головна мета таких засобів - продемонструвати, як кожен процес перетворює свої вхідні дані у вихідні, а також виявити відносини між цими процесами.

Першим кроком при побудові ієрархії DFD є побудова контекстних діаграм 0-го рівня.

Для складних ІС будується ієрархія контекстних діаграм. При цьому контекстна діаграма 1-го рівня містить набір підсистем, з'єднаних потоками

даних. Контекстні діаграми наступного рівня деталізують контекст і структуру підсистем до рівня окремих завдань.

Для кожної підсистеми, присутньої на контекстних діаграмах, виконується її деталізація за допомогою діаграми DFD. Кожен процес, в свою чергу, може бути деталізований за допомогою окремої діаграми або міні специфікації.

Для створення особистого кабінету спроектуємо Data Flow діаграми 0-го і 1-го рівнів (рисунок 2.2 і рисунок 2.3), дотримуючись наступних правил: правила балансування, правила нумерації і правила семи.

Побудова діаграми потоків даних 0-го рівня

На рисунку 2.2 ми зобразили головний процес - «Управління телекомунікаційними послугами та рахунками». Цей процес являє собою перетворення вхідних потоків даних у вихідні відповідно до певного алгоритму і бізнес-логікою підприємства. Він описує основні функції проектованої інформаційної системи.

Дві зовнішні сутності «Клієнт» і «Підприємство» є об'єктами, які є джерелом або приймачем системних даних.

Званими стрілками ми проілюстрували потоки даних, які є в нашому макеті абстракціями, використовуваними для передачі інформації з однієї частини системи в іншу.

Рисунок 2.2 Data Flow діаграма 0-го рівня

Побудова діаграми потоків даних 1-го рівня

Основний процес можна деталізувати, розбивши його на кілька окремих специфічних підпроцесів. Для цього ми створили Data Flow діаграму 1 рівня (рисунок 2.3).

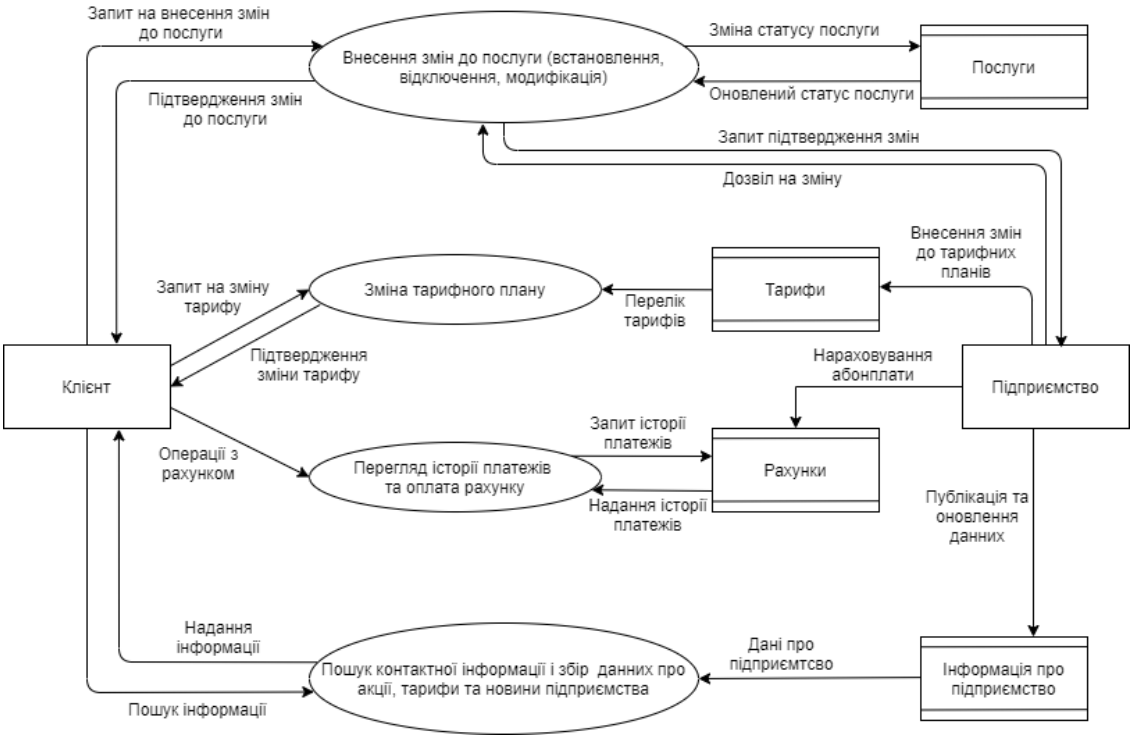


Рисунок 2.3 Діаграма потоків даних 1-го рівня

На діаграмі потоків даних 1-го рівня ми більш детально описали функції інформаційної системи. Дані функції були зображені на діаграмі у вигляді додаткових процесів:

- внесення змін до послуг (установка, відключення, модифікація);
- зміна тарифного плану;
- перегляд історії платежів і оплата рахунку;
- пошук контактної інформації, новини підприємства і збір даних про акції і тарифи.

Згідно з правилами побудови DF-діаграм ми виділили сховища даних «Рахунки», «Послуги», «Тарифи» і «Інформація про підприємство», які дозволять нам на зазначених ділянках визначати дані, які будуть зберігатися в пам'яті між процесами. У цих абстрактних сутностей зберігається

інформація, необхідна для функціонування проектованої інформаційної системи.

Зв'язок між згаданими вище елементами діаграми буде здійснюватися за допомогою потоків даних, зображених на діаграмі у вигляді іменованих стрілок.

2.1.2 Логічна структура бази даних

Кожен факт, що зберігається в БД, повинен зберігатися один-єдиний раз, оскільки дублювання може привести до неузгодженості між копіями однієї і тієї ж інформації.

Нормалізацією схеми бази даних називається процедура, що виконується над базою даних з метою видалення в ній надмірності.

Виділяються шість нормальних форм, п'ять з яких так і називаються: перша, друга, третя, четверта, п'ята нормальна форма, а також нормальна форма Бойса-Кодда, що лежить між третьою і четвертою.

База даних вважається нормалізованою, якщо її таблиці представлені як мінімум в третій нормальній формі.

Алгоритм приведення ненормалізованих схем в 3НФ показаний на рисунку 2.4. На практиці побудови третьої нормальної форми в більшості випадків достатньо і на цьому процес побудови реляційної БД закінчується.

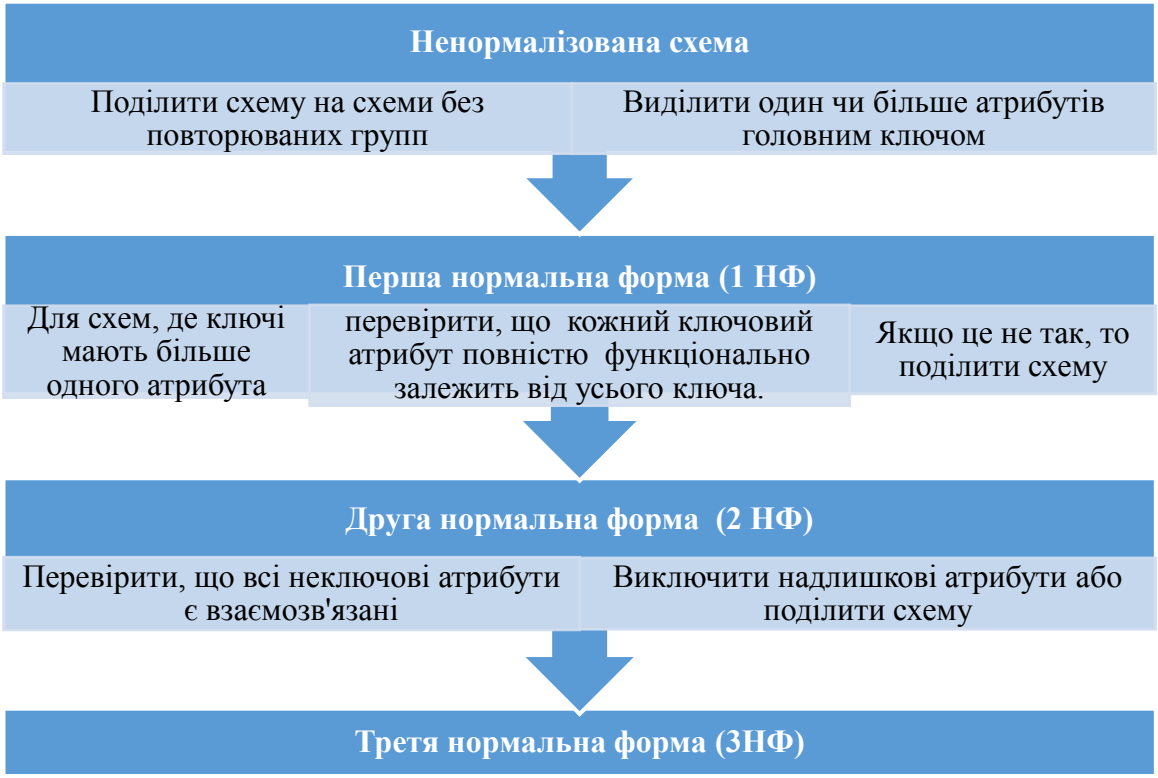


Рисунок 2.4 Алгоритм приведення ненормалізованих схем в 3НФ

Зобразимо отримані відношення і їх зв’язки за допомогою ER-діаграм (рисунок 2.5 і Додаток 1).

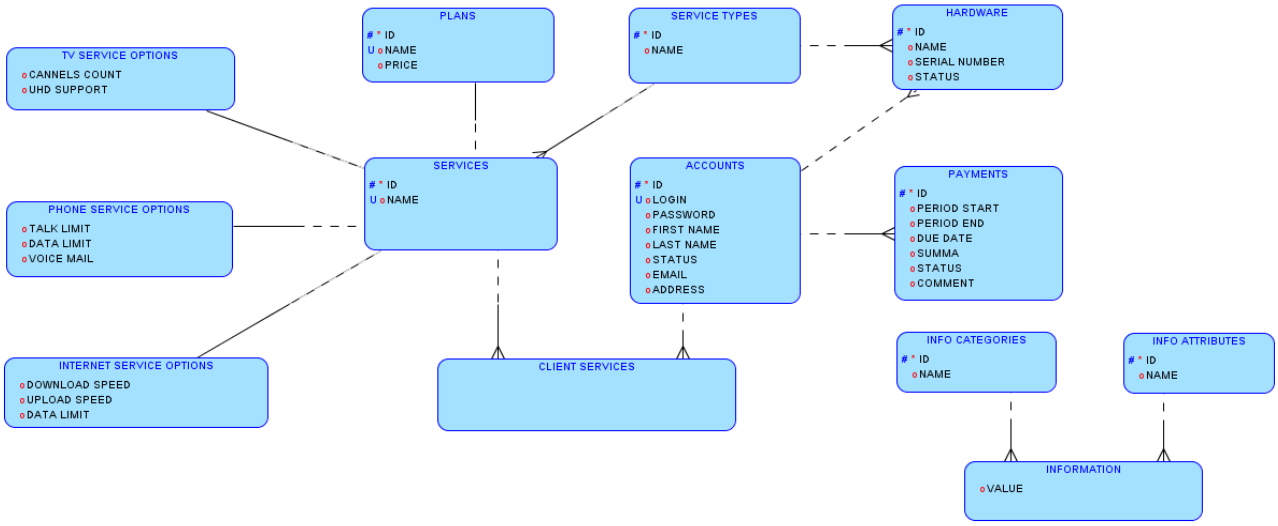


Рисунок 2.5 ER-діаграма. Логічна модель.

Розглянемо структуру БД, імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження (Таблиця 2.1).

Таблиця 2.1 Структура БД

Таблиця	Стовпець	Зміст	Тип даних	Ключовий	Обмеження
accounts	id	Унікальний номер аккаунта користувача	NUMBER(10)	PK	NOT NULL
	login	Логін користувача	VARCHAR2(100)		NOT NULL
	password	Пароль користувача	VARCHAR2(100)		
	first_name	Ім'я користувача	VARCHAR2(100)		
	last_name	Прізвище користувача	VARCHAR2(100)		
	status	Поточний статус аккаунта	VARCHAR2(100)		NOT NULL, можливі значення: 'Active', 'Inactive'
	email	Адреса електронної пошти користувача	VARCHAR2(100)		
	address	Фізична адреса користувача	VARCHAR2(1000)		
plans	id	Унікальний номер тарифного плану	NUMBER(10)	PK	NOT NULL
	name	Назва тарифного плану	VARCHAR2(100)		NOT NULL
	price	Місячна абонплата (грн.)	FLOAT		Значення>= 0
service_types	id	Унікальний номер типу сервісу	NUMBER(10)	PK	NOT NULL
	name	Назва типу сервісу (інтернет, телебачення, телефонія)	VARCHAR2(200)		NOT NULL
services	id	Унікальний номер сервісу	NUMBER(10)	PK	NOT NULL
	name	Назва сервісу (інтернет,	VARCHAR2(200)		NOT NULL

		телебачення, телефонія)			
	type_id	Номер типу сервісу	NUMBER(10)	PFK	NOT NULL
	plan_id	Номер тарифного плану	NUMBER(10)	PFK	NOT NULL
client_services	account_id	Номер аккаунта користувача, до якого підключено сервіс	NUMBER(10)	PFK	NOT NULL
	service_id	Номер підключеного сервісу	NUMBER(10)	PFK	NOT NULL
payments	id	Унікальний номер платежу	NUMBER(10)	PK	NOT NULL
	account_id	Номер аккаунта користувача, для якого призначений платіж	NUMBER(10)	PFK	NOT NULL
	period_start	Дата початку розрахункового періоду	DATE		NOT NULL, period_start <= period_end
	period_end	Дата кінця розрахункового періоду	DATE		NOT NULL, period_start <= period_end
	due_date	Кінцева дата платежу	DATE		NOT NULL
	summa	Сума до оплати (грн.)	FLOAT		NOT NULL
	status	Статус платежу (сплачено / не сплачено)	VARCHAR2(50)		NOT NULL, можливі значення: 'Paid', 'Not Paid'
hardware	id	Унікальний номер платежу	NUMBER(10)	PK	NOT NULL
	account_id	Номер аккаунта користувача, до якого прив'язана	NUMBER(10)	FK	

		обладнання			
	service_type_id	Номер типу сервісу, для якого призначене обладнання	NUMBER(10)	FK	NOT NULL
	name	Назва обладнання	VARCHAR2(200)		NOT NULL
	serial_number	Серійний номер обладнання	VARCHAR2(200)		NOT NULL
	status	Статус обладнання (активний / неактивний).	VARCHAR2(50)		NOT NULL, можливі значення: 'Active', 'Inactive'
info_categories	id	Унікальний номер категорії інформації	NUMBER(10)	PK	NOT NULL
	name	Назва категорії інформації	VARCHAR2(200)		NOT NULL
info_attributes	id	Унікальний номер параметра інформації	NUMBER(10)	PK	NOT NULL
	name	Ім'я параметра	VARCHAR2(200)		NOT NULL
information	attr_id	Номер параметра інформації	NUMBER(10)	PFK	NOT NULL
	category_id	Номер категорії інформації	NUMBER(10)	PFK	NOT NULL
	value	Значення параметру	VARCHAR2(2000)		NOT NULL
internet_service_options	service_id	Номер інтернет сервісу	NUMBER(10)	PFK	NOT NULL
	download_speed	Швидкість скачування даних (Мб /с)	VARCHAR2(20)		NOT NULL
	upload_speed	Швидкість завантаження даних (Мб /с)	VARCHAR2(20)		NOT NULL
	data_limit	Сумарний місячний ліміт	VARCHAR2(20)		NOT NULL

		трафіку (ГБ)			
tv_service_options	service_id	Номер телевізійного сервісу	NUMBER(10)	PFK	NOT NULL
	channels_count	Кількість підключених каналів	NUMBER(10)		NOT NULL
	uhd_support	Підтримка UHD (стандарт надвисокої чіткості)	VARCHAR2(10)		NOT NULL, можливі значення: 'On', 'Off'
phone_service_options	service_id	Номер аккаунта користувача, до якого прив'язано обладнання	NUMBER(10)	PFK	NOT NULL
	talk_limit	Місячний ліміт тривалості розмови (хв)	NUMBER(10)		NOT NULL
	data_limit	Місячний ліміт даних для мобільного інтернет (ГБ)	NUMBER(10)		NOT NULL
	voice_mail	Наявність голосової пошти	VARCHAR2(5)		NOT NULL, можливі значення: 'On', 'Off'

2.2 Фізична реалізація бази даних

В якості СУБД для бази даних було вирішено вибрати СУБД Oracle. Дане рішення базується на наступних факторах:

- масштабованість;
- висока продуктивність;
- підтримка 24/7;
- надійність (налагодженість процедур резервного копіювання та відновлення)
- велика спільнота програмістів і доступність документації.

Одним із суттєвих недоліків СУБД Oracle є те, що даний продукт в основному є платним. Але у Oracle також є безкоштовна редакція СУБД Oracle Database. Це редакція eXpress Edition (XE). Вона доступна для скачування на сайті https://www.oracle.com/ безкоштовно для використання в бізнесі і має версії під Windows і Linux. Якщо безкоштовна версія в майбутньому перестане задовольняти зростаючі потреби бізнесу, тоді є можливість з мінімальними витратами часу і ресурсів перейти на комерційну версію Oracle СУБД, яка пропонує повний функціонал.

2.2.1 Створення ключових сутностей бази даних

Створення таблиць

Основними сутностями в БД є таблиці. Нижче наведено SQL сценарій створення таблиць, ініціалізація обмежень і зв'язків між таблицями.

Синтаксис створення таблиці в Oracle SQL вказано нижче. Більш детальну інформацію по створенню таблиць і інших об'єктів можна знайти на сайті Oracle.

```
CREATE TABLE table_name
(
  column1 datatype [ NULL | NOT NULL ],
  column2 datatype [ NULL | NOT NULL ],
  ...
  column_n datatype [ NULL | NOT NULL ]
  [ CONSTRAINT ]
);
```

Параметри і аргументи:

table_name - ім'я створюваної таблиці,

column1, column2, ... column_n - імена стовпців в створюваній таблиці. Кожен стовпець повинен мати тип даних (datatype). Стовпець повинен бути визначений як «null» або «not null», і якщо це значення залишається порожнім, база даних передбачає «null» як значення за замовчуванням.

SQL-код для створення таблиць для нашої БД надано у Додатку 3.

Створення послідовностей і тригерів для генерації первинних ключів

Більшість з таблиць, створених вище мають первинний ключ, який повинен генеруватися автоматично при додаванні запису в таблицю. Для цього в Oracle SQL використовуються послідовності. Синтаксис створення послідовності наведено нижче.

```
CREATE SEQUENCE sequence_name
  MINVALUE value1
  MAXVALUE value2
  START WITH value3
  INCREMENT BY value4
  CACHE value5;
```

Параметри і аргументи:

sequence_name - ім'я створюваної послідовності,

value1 – позначає мінімальне значення послідовності,

value2 – позначає максимальне значення послідовності,

value3 – позначає значення, з якого починається відлік послідовності,

value4 – позначає інтервал між наступними номерами. Це цілочисельне значення може бути будь-яким позитивним або негативним цілим числом, але не може бути 0,

value5 – позначає скільки значень послідовності база даних попередньо розподіляє і зберігає в пам'яті для більш швидкого доступу.

Після створення послідовності, чергове її значення можна отримати наступним оператором:

```
sequence_name.NEXTVAL;
```

Для того щоб при вставці записів в таблицю цей оператор викликався автоматично, створимо тригер за допомогою діалекту Oracle PL / SQL.

Існує кілька типів тригерів в залежності від дії, за яким даний тригер спрацьовує:

- BEFORE INSERT

- AFTER INSERT
- BEFORE UPDATE
- AFTER UPDATE
- BEFORE DELETE
- AFTER DELETE

Для відстеження вставки записів в таблицю і генерації нового первинного ключа ми будемо використовувати тригер типу BEFORE INSERT.

Його синтаксис описаний нижче:

```
CREATE [ OR REPLACE ] TRIGGER trigger_name
BEFORE INSERT
  ON table_name
  [ FOR EACH ROW ]

DECLARE
  -- variable declarations

BEGIN
  -- trigger code

EXCEPTION
  WHEN ...
  -- exception handling

END;
```

Параметри і аргументи:

trigger_name - ім'я створюваного тригера,

BEFORE INSERT - цей параметр вказує що тригер спрацює перед операцією вставки даних в таблицю,

table_name - ім'я таблиці, для якої створено тригер.

В якості виклику коду в тригері ми будемо викликати наступний код, який повертає чергове значення послідовності окремо для кожної таблиці:

```
SELECT sequence_name. NEXTVAL
   INTO :new.id
   FROM dual;
```

Повна версія скрипта для створення послідовностей і тригерів надана у Додатку 4.

Після того як ключові сутності БД створені, необхідно застосувати зміни, виконавши команду COMMIT. Потім потрібно заповнити створені таблиці даними.

2.3 Проектування веб-додатку

2.3.1 Опис архітектури веб-додатку

Так як розробляємий нами додаток «Особистий кабінет» є веб-додатком, то необхідно розібрати суть цього поняття і його ключові моменти.

Веб-додаток - клієнт-серверний додаток, в якому споживач взаємодіє з сервером за допомогою браузера, а за сервер відповідає веб-сервер. Логіка веб-додатку розподілена між сервером і клієнтом, зберігання даних здійснюється, переважно, на сервері, обмін інформацією відбувається по мережі. Одним з переваг такого підходу є той факт, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є міжплатформенними службами.

Веб-додаток складається з клієнтської і серверної частин, тим самим реалізуючи технологію «клієнт-сервер».

Клієнтська частина реалізує користувальницький інтерфейс, формує запити до сервера і обробляє відповіді від нього.

Серверна частина отримує запит від клієнта, виконує обчислення, після цього формує веб-сторінку і відправляє її клієнту через мережу з використанням протоколу HTTP.

Саме веб-додаток може виступати в якості клієнта інших служб, наприклад, бази даних або іншого веб-додатки, розташованого на іншому сервері (рисунок 2.6).

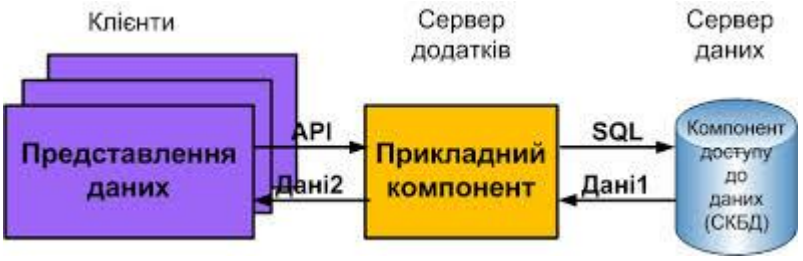


Рисунок 2.6 Схема функціонування веб-додатку

Основною мовою програмування для реалізації програми «Особистий кабінет» було обрано мову Java, що включає набір специфікацій Java Platform, Enterprise Edition (Java EE).

Платформа J2EE пристосована для розробки багаторівневих Web-додатків. При роботі з такими додатками користувач формує свої запити, заповнюючи HTML-форми в браузері, який упаковує їх в HTTP-повідомлення і пересилає Web-серверу. Web-сервер передає ці повідомлення Web-компонентів, що виділяють з них вихідні запити користувача і передає їх для обробки компонентів EJB. Результати роботи EJB компонентів перетворюються Web-компонентами в динамічно генеруються HTML-сторінки, і відправляються назад користувачеві, постаючи перед ним у вікні браузера. Аплети використовуються там, де потрібен більш функціональний інтерфейс, ніж стандартні форми і сторінки HTML (рисунок 2.7).

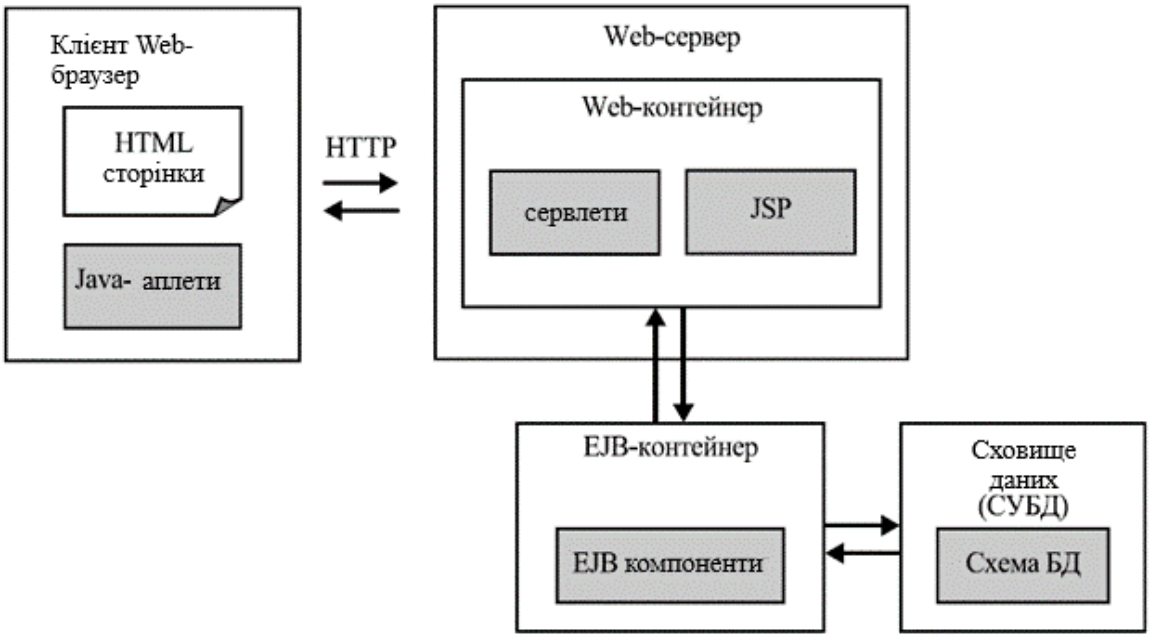


Рис. 2.7 Типова архітектура J2EE-додатку.

2.3.2 Створення UML-діаграм

Центральне місце в об’єктно-орієнтованому проектуванні займає розробка логічної моделі системи у вигляді діаграми класів. Для її створення прийнято використовувати мову моделювання UML.

UML (Unified Modeling Language - уніфікована мова моделювання) - мова графічного опису для об'єктного моделювання в області розробки програмного забезпечення, моделювання бізнес-процесів, системного проектування та відображення організаційних структур.

Нотація UML надає широкі можливості для відображення додаткової інформації (абстрактні операції і класи, стереотипи, загальні і приватні методи, деталізовані інтерфейси, параметризовані класи). При цьому можливе використання графічних зображень для асоціацій та їх специфічних властивостей, таких як ставлення агрегації, коли складовими частинами класу можуть виступати інші класи.

Діаграма класів складається з множини елементів, які в сукупності відображають декларативні знання про предметну область. Ці знання інтерпретуються в базових поняттях мови UML, таких як класи, інтерфейси і відносини між ними і їх складовими компонентами. При цьому окремі компоненти цієї діаграми можуть утворювати пакети для представлення більш загальної моделі системи. Якщо діаграма класів є частиною деякого пакета, то її компоненти повинні відповідати елементам цього пакета, включаючи можливі посилання на елементи з інших пакетів.

Діаграму класів додатку «Особистий кабінет» представлено у Додатку 2.

Діаграма послідовності - діаграма, на якій для деякого набору об'єктів на єдиній тимчасової осі показаний життєвий цикл будь-якого певного об'єкта (створення-діяльність-знищення якоїсь сутності) і взаємодія акторів (дійових осіб) ІС в рамках якого-небудь певного прецеденту (відправка запитів і отримання відповідей).

Основними елементами діаграми послідовності є позначення об'єктів (прямокутники з назвами об'єктів), вертикальні «лінії життя», що відображають плин часу, прямокутники, що відображають діяльність об'єкта або виконання ним певної функції (прямокутники на пунктирній «лінії життя»), і стрілки, що показують обмін сигналами або повідомленнями між об'єктами. На даній діаграмі об'єкти розташовуються зліва направо.

На діаграмі нижче (рисунок 2.8) показаний процес управління сервісами. Коли клієнт вибирає опцію установки/модифікації або відключення послуги в додатку «Особистий кабінет», надсилається повідомлення в зовнішні системи (External Inventory, Billing) для активації обладнання та розрахунку платежів відповідно. Після завершення цих процесів і отримання відповіді від цих систем, поточний статус сервісу записується в базу даних програми і оновлюється на сторінці користувача.

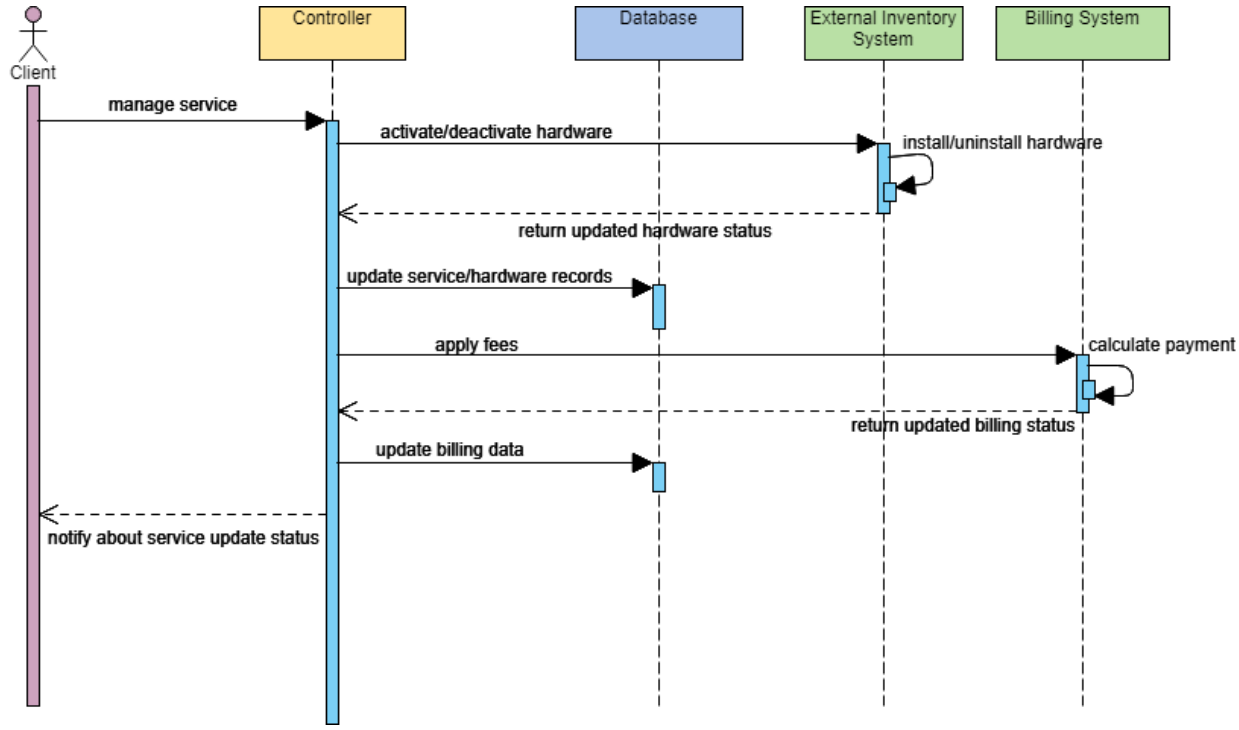


Рисунок 2.8 Діаграма послідовності. Управління сервісами.

На наступній діаграмі послідовності (рисунок 2.9) зображені основні функції, які доступні користувачам програми: перевірка статусу послуг, отримання інформації про баланс і історії платежів, отримання інформації про продукти компанії.

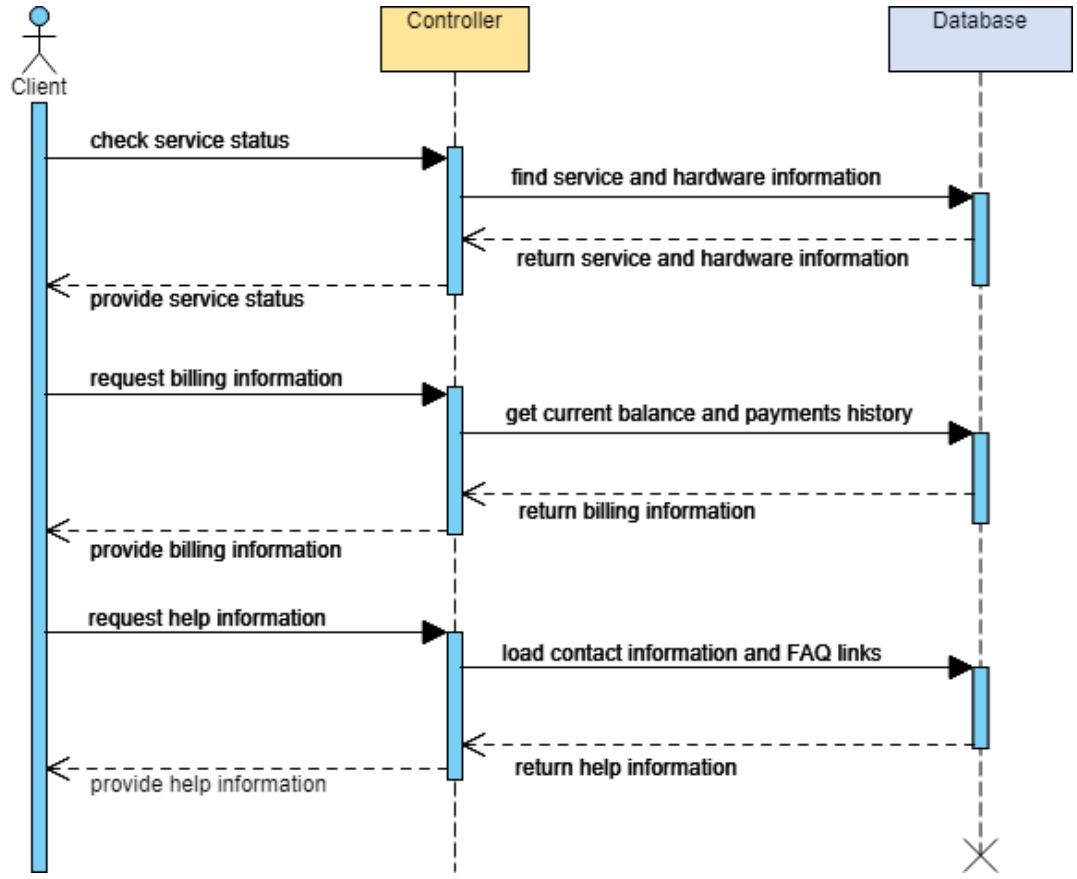


Рисунок 2.9 Діаграма послідовності. Основні функції додатку.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Створення й настройка додатку

Для створення веб-додатку необхідно обрати відповідні засоби розробки. До їх числа належать такі засоби як: середовище для написання коду, для розробки бази даних, вибір серверу й системи автоматичної збірки.

3.1.1 Вибір середовища розробки для створення додатку

Інтегроване середовище розробки (Integrated development environment, IDE) - це програма, яка містить в собі інструменти для розробки програмного забезпечення.

Середовище розробки для роботи з базою даних

Для роботи з SQL скриптами і внесення змін в БД була використана програма Oracle SQL Developer.

Oracle SQL Developer - це безкоштовне інтегроване середовище розробки, що спрощує створення і управління базами даних. SQL Developer пропонує повну наскрізну розробку PL/SQL-додатків, робочий лист для запуску запитів і сценаріїв, консоль адміністратора баз даних для управління базою даних, інтерфейс звітів, повне рішення для моделювання даних і платформу міграції для переміщення сторонніх баз даних (рисунок 3.1).

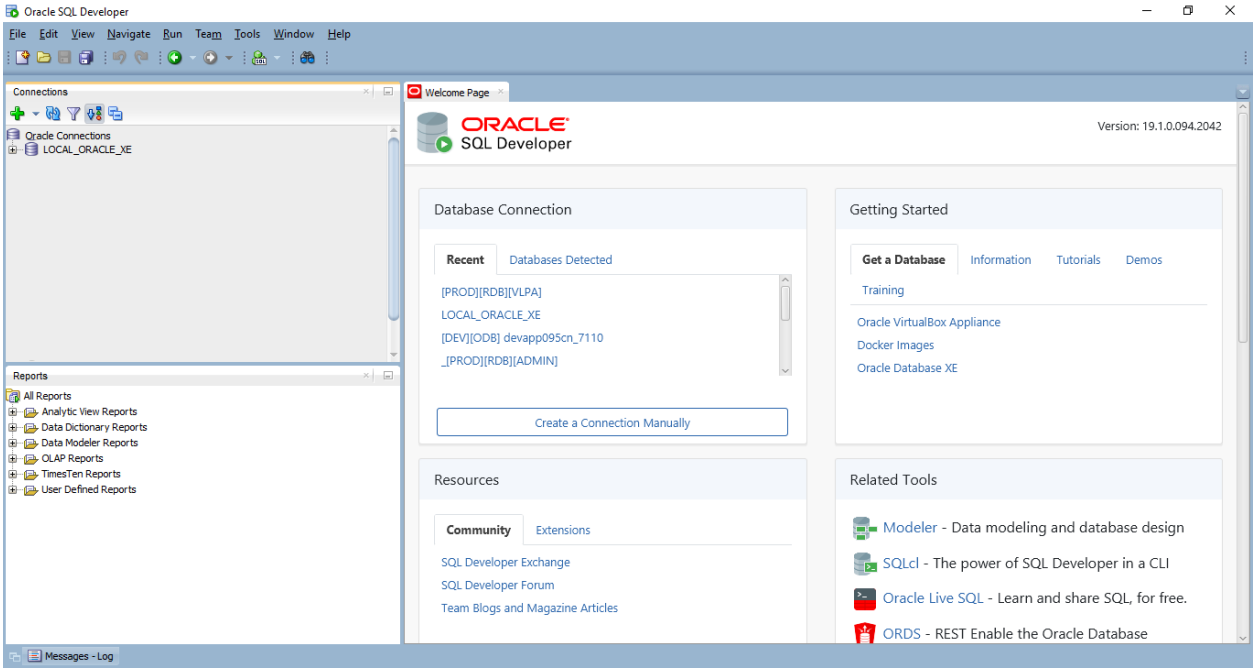


Рисунок 3.1. Інтерфейс програми Oracle SQL Developer

Корпорація Oracle надає продукт безкоштовно. Саме середовище написано на мові програмування Java, працює на всіх платформах, де доступна середовище розробки Java SE.

Середовище розробки для Java

Сучасне середовище розробки на мові Java включає в себе:

- текстовий редактор з підсвічуванням коду;
- компілятор або інтерпретатор;
- браузер класів, інспектор об’єктів і діаграму ієрархії классов;
- засоби автоматизації збірки;
- відладчик;
- кошти для інтеграції з системами управління версіями;
- інструменти для спрощення конструювання графічного інтерфейсу користувача.

Основними IDE для написання коду на мові програмування Java є IntelliJ IDEA, Eclipse, NetBeans, JDeveloper.

Наш вибір зупинився на IntelliJ IDEA (рисунок 3.2), так як вона надає безліч інструментів для продуктивної роботи і ідеально підходить для створення корпоративних, мобільних і веб-додатків.

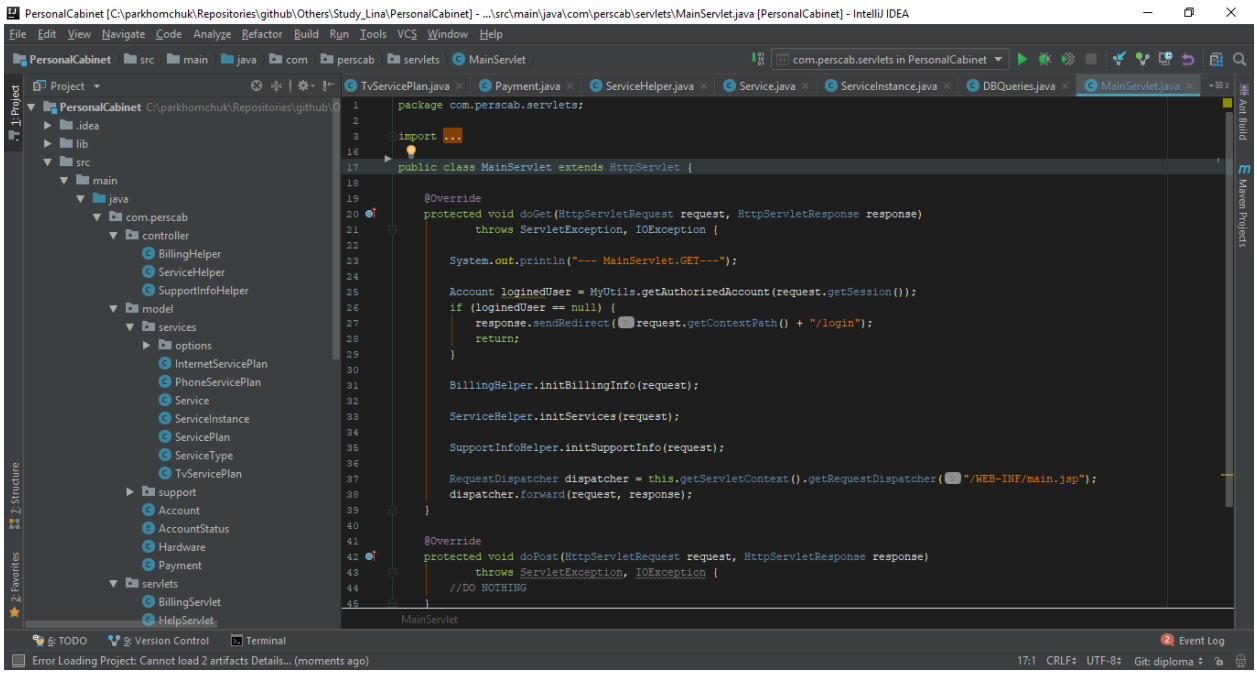


Рисунок 3.2 Інтерфейс середовища розробки IntelliJ IDEA

IntelliJ IDEA поширюється в двох версіях, одна з яких безкоштовна - Free Community Edition. У платній версії є підтримка фреймворків Spring (Spring MVC framework, Spring Security, Spring Boot, Spring Integration і подібні), Node.js, Angular React, Grails, можливість використовувати додаткові мови (javascript, typescript, coffeescript) і взаємодіяти майже з семи популярними серверами (Tomcat, TomEE, GlassFish, JBoss, WildFly, Weblogic, WebSphere, Geronimo, Virgo і т. д.).

Для реализаци нашого додатку буде досить Free Community Edition версії для IntelliJ IDEA, так як вона містить всі необхідні засоби для розробки проекту.

3.1.2 Налаштування системи збірки Maven

Maven - інструмент для автоматизації збирання проектів на основі опису їх структури в файлах на мові POM (англ. Project Object Model), що є підмножиною XML. Даний інструмент використовується в розробці додатків

на Java, хоча є плагіни для інтеграції з C / C ++, Ruby, Scala, PHP і іншими мовами.

Одна з головних особливостей фреймворка - декларативний опис проекту. Це означає, що розробнику не потрібно приділяти увагу кожному аспекту збірки - всі необхідні параметри налаштовані за замовчуванням. Зміни потрібно вносити лише в тому обсязі, в якому програміст хоче відхилитися від стандартних налаштувань.

Ще одна перевага проекту - гнучке управління залежностями. Maven вміє довантажувати в свій локальний репозиторій сторонні бібліотеки, вибирати необхідну версію пакету, обробляти транзитивні залежності.

Ми будемо працювати з Maven за допомогою командного рядка (рисунок 3.3), однак цей фреймворк також інтегрований в Eclipse, IntelliJ IDEA, NetBeans і інші IDE.

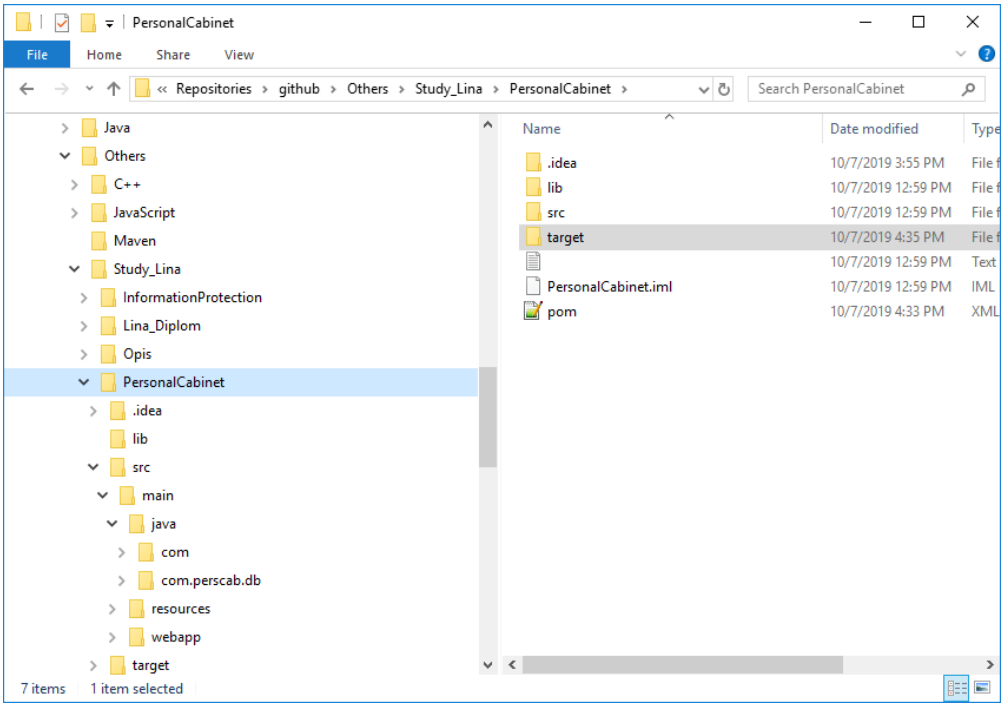


Рисунок 3.3 Структура Maven проекту додатку «Особистий кабінет»

Конфігураційний POM файл для застосування «Особистий кабінет» буде виглядати наступним чином (рисунок 3.4):

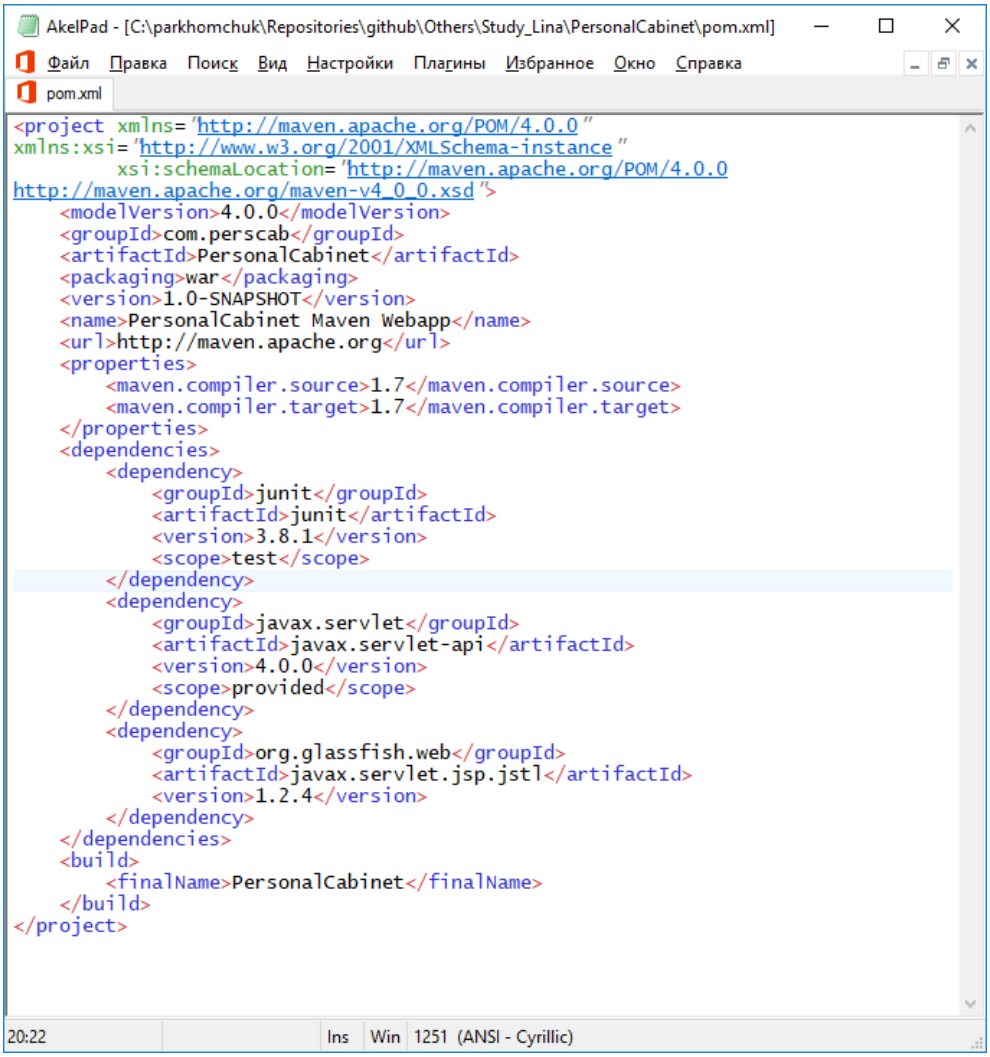


Рисунок 3.4 Конфігураційний POM файл

Запуск збірки проекту здійснюється командою "mvn clean install". Опис життєвого циклу збірки, команд і доступних параметрів можна знайти на офіційно сайті Apache Maven.

Після успішного завершення буде створений файл PersonalCabinet.war, який по суті є веб-архівом з усіма необхідними ресурсами, який потрібно буде скопіювати на веб-сервер (рисунок 3.5).

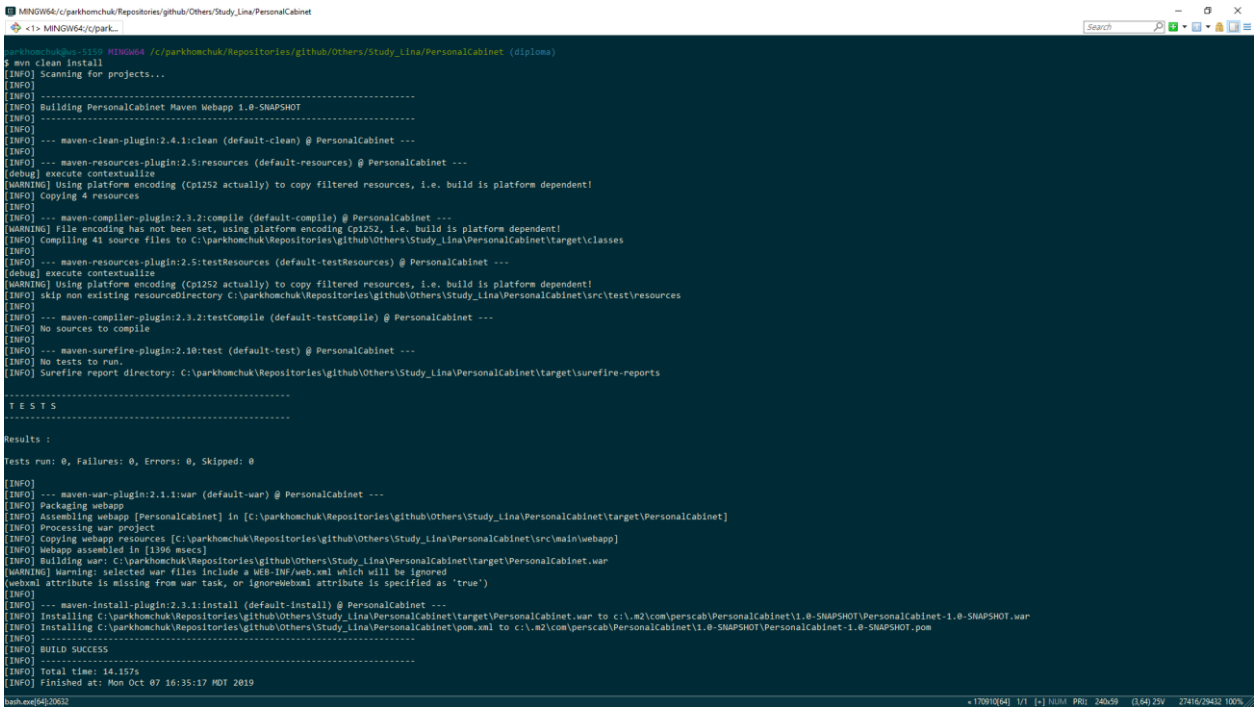


Рисунок 3.5 Результат збірки Maven проекту

3.1.3 Використання системи контролю версій

Система контролю версій (від англ. Version Control System, VCS або Revision Control System) - програмне забезпечення для полегшення роботи зі змінною інформацією. Система контролю версій (СКВ) дозволяє зберігати кілька версій одного і того ж документа, при необхідності повертатися до попередніх версій, визначати, хто і коли зробив ту чи іншу зміну, надає можливості для створення нових і злиття існуючих гілок проекту, виробляє контроль доступу користувачів до проекту.

Локальні системи контролю версій

Багато хто віддає перевагу контролювати версії, просто копіюючи файли в інший каталог. Такий підхід дуже поширений, тому що простий, але він і частіше дає збої. Щоб вирішити цю проблему, були розроблені локальні СКВ з простою базою даних, в якій зберігаються всі зміни потрібних файлів (рисунок 3.6).

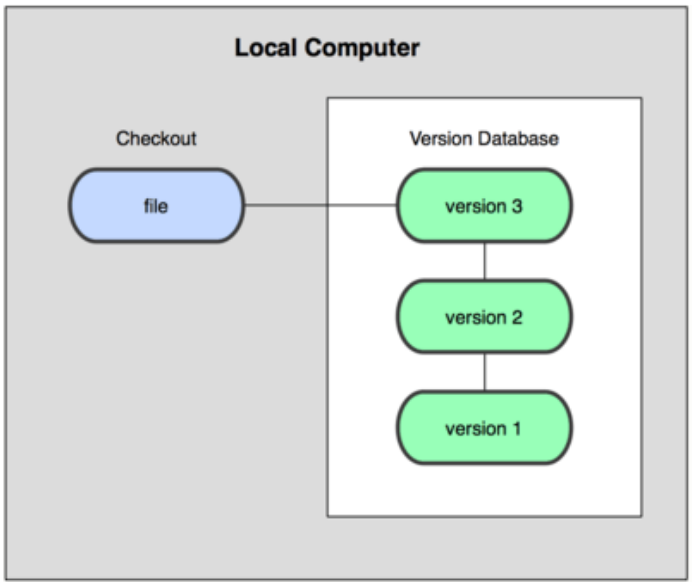


Рисунок 3.6 Схема локальної СКВ.

Централізовані системи контролю версій

Наступною основною проблемою є необхідність співпрацювати з розробниками за іншими комп'ютерами. Щоб розв'язати цю проблему, були створені централізовані системи контролю версій (ЦСКВ). У таких системах, наприклад, CVS, Subversion і Perforce, є центральний сервер, на якому зберігаються всі файли під версійність контролем, і ряд клієнтів, які отримують копії файлів з нього (рисунок 3.7).

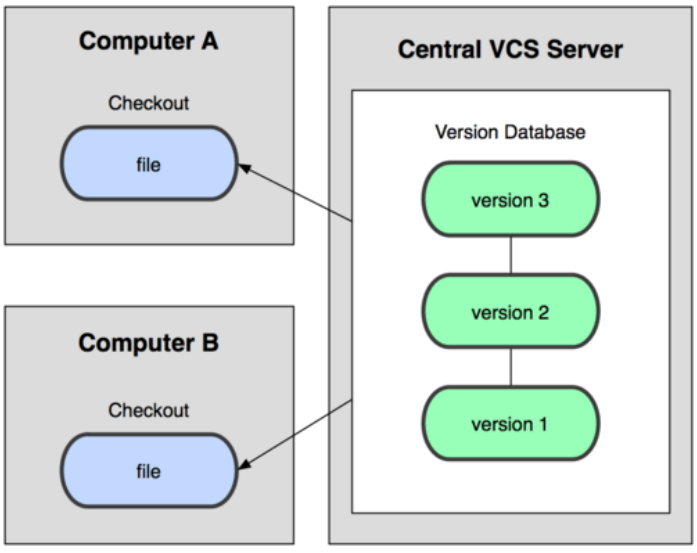


Рисунок 3.7 Схема централізованого контролю версій.

Однак при такому підході є й кілька серйозних недоліків. Найбільш очевидний - централізований сервер є вразливим місцем всієї системи. Якщо

сервер вимикається на годину, то протягом години розробники не можуть взаємодіяти, і ніхто не може зберегти нової версії своєї роботи. Якщо ж пошкоджується диск з центральною базою даних і немає резервної копії, можна втратити абсолютно все. Локальні системи контролю версій схильні до тієї ж проблеми: якщо вся історія проекту зберігається в одному місці, то є ризик втратити все.

Розподілені системи контролю версій

Описані вище проблеми вирішуються за допомогою розподілених систем контролю версій (РСКВ). У таких системах як Git, Mercurial, Bazaar або Darcs клієнти не просто вивантажують останні версії файлів, а повністю копіюють весь репозиторій (рисунок 3.8).

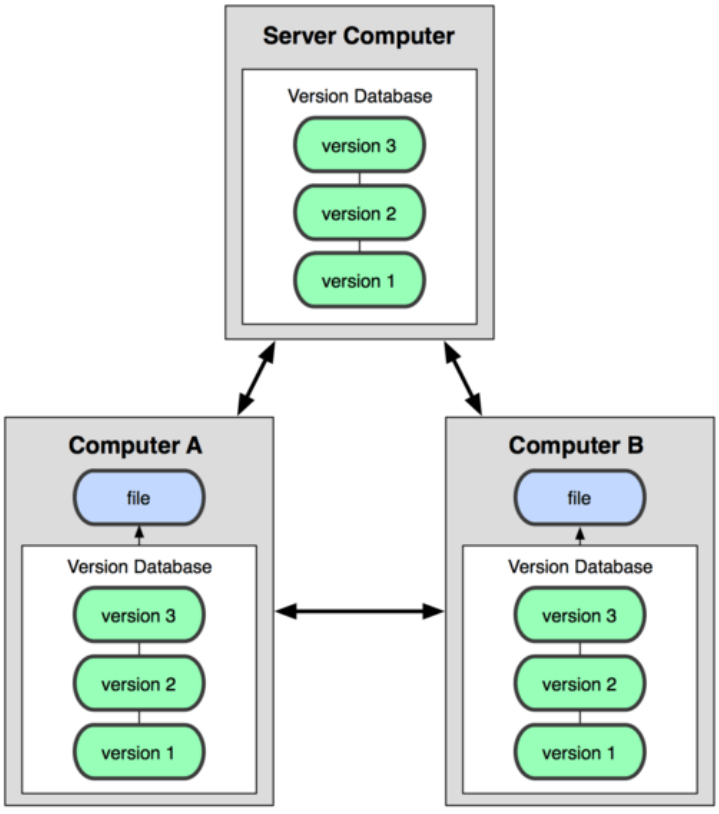


Рисунок 3.8 Схема розподіленої системи контролю версій.

Крім того, в більшій частині цих систем можна працювати з декількома віддаленими репозиторіями, таким чином, можна одночасно працювати по-різному з різними групами людей в рамках одного проекту. Так, в одному

проекті можна одночасно вести кілька типів робочих процесів, що неможливо в централізованих системах.

Для додатка «Особистий кабінет» будемо використовувати одну з найпопулярніших РСКВ - Git.

В якості репозиторія був обраний GitHub (рисунок 3.9). GitHub - найбільший веб-сервіс для хостингу IT-проектів і їх спільної розробки. Сервіс безкоштовний для проектів з відкритим вихідним кодом і (з 2019 роки) невеликих приватних проектів, надаючи їм всі можливості (включаючи SSL), а для великих корпоративних проектів пропонуються різні платні тарифні плани.

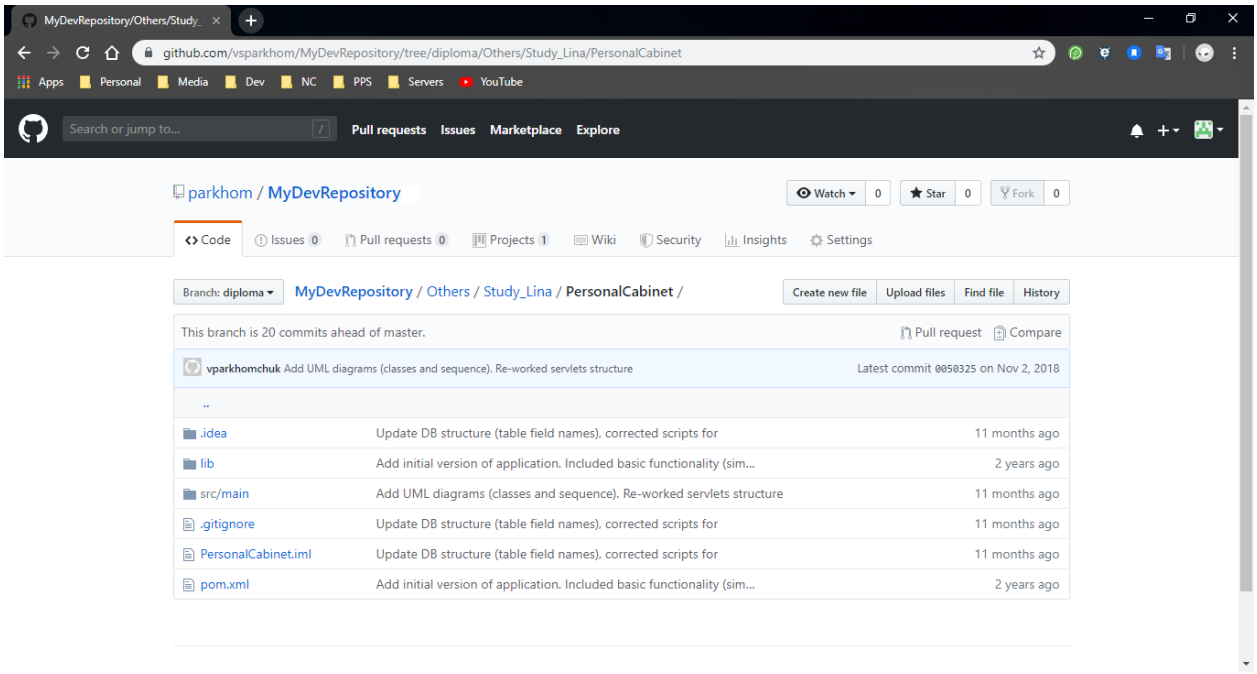


Рисунок 3.9 Репозиторій GitHub

Далі в потрібно створити робочу копію сховища та створити окрему гілку для змін (рисунок 3.10). При подальшій розробці проекту всі зміни будуть зберігатися в цю гілку.

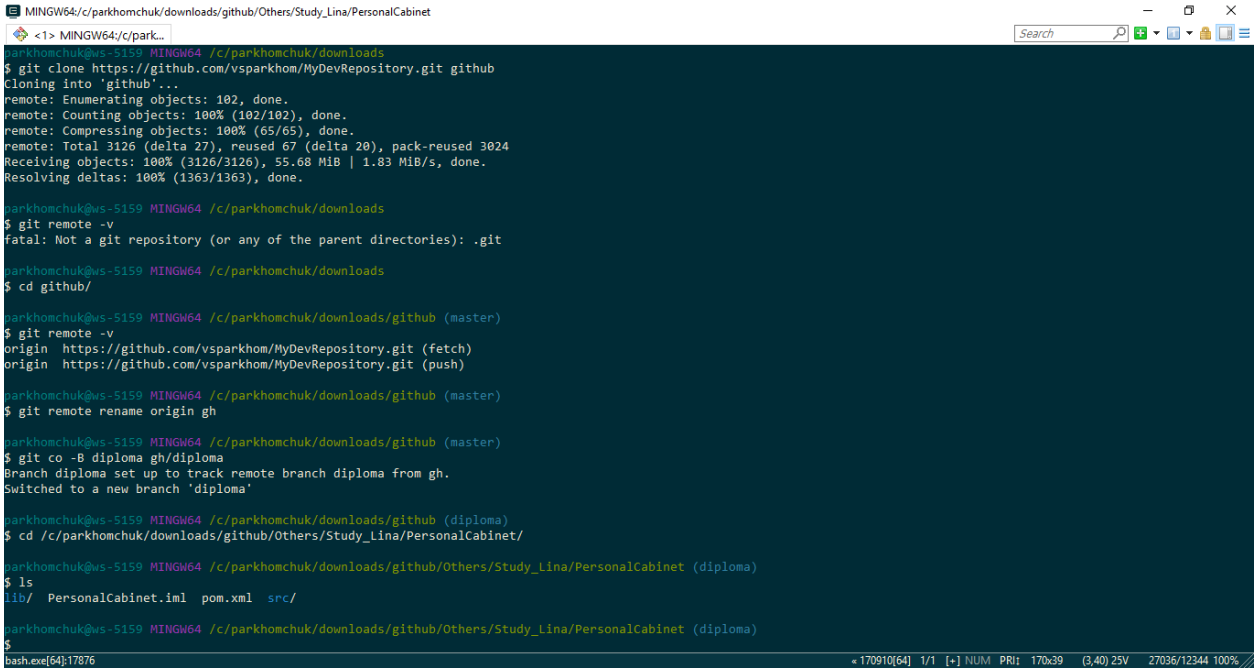


Рисунок 3.10 Створення та налагодження робочої копії сховища

3.1.4 Налаштування веб-сервера і розгортання додатку

Після успішної збірки вихідного коду необхідно розгорнути додаток на тестовий веб-сервер для подальшого тестування і налагодження. Як веб-сервера буде обраний Apache Tomcat.

Apache Tomcat - контейнер сервлетів з відкритим вихідним кодом, що розробляється Apache Software Foundation. Реалізує специфікацію сервлетів, специфікацію JavaServer Pages (JSP) і JavaServer Faces (JSF). Написаний на мові Java. Він дозволяє запускати веб-додатки і містить ряд програм для самоконфігурації.

Останню версію Apache Tomcat можна скачати на офіційному сайті.

Після настройки сервера (конфігурація портів, створення користувачів і відповідних ролей) потрібно запустити сервер командою ./startup.sh. Потім потрібно в браузері відкрити адресу http: // localhost: 8080 / і з'явиться стандартне вікно сервера.

Кнопка «Server Status» дозволяє подивитися на поточний статус сервера.

Натиснувши на посилання «Manager App» можна перейти на сторінку управління веб-додатками (рисунок 3.11).

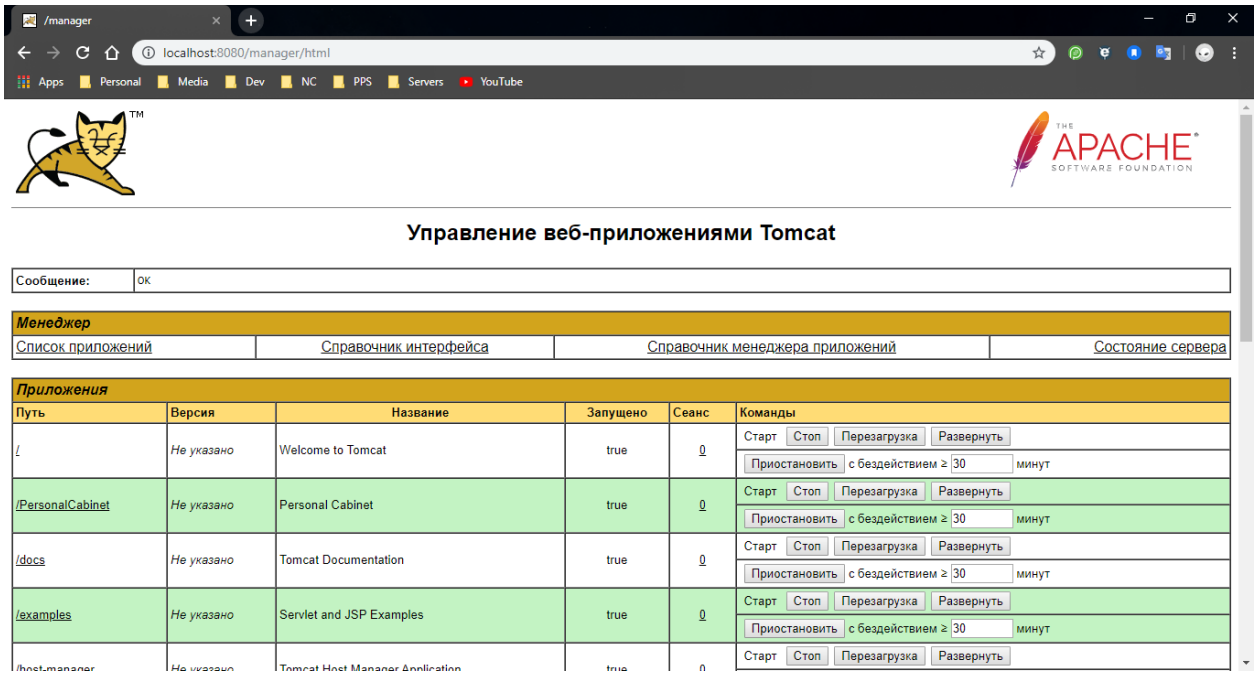


Рисунок 3.11 Управління веб-додатками Apache Tomcat

На цій же сторінці доступні функції для старту, зупинки і перезавантаження встановлених додатків.

Також, тут присутня можливість розгортання програми. Для цього необхідно вказати основні параметри (веб-адреса в браузері і шлях до WAR файлу) і натиснути кнопку «Розгорнути» (рисунок 3.12).

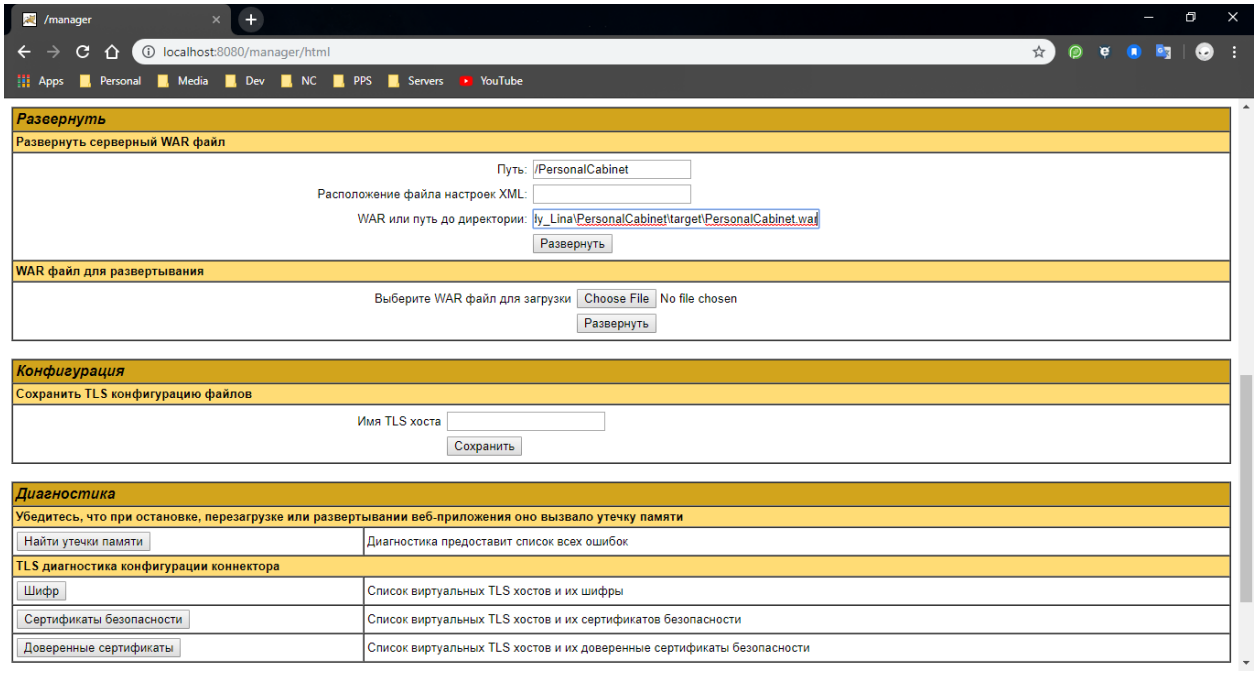


Рисунок 3.12 Сторінка для розгортання додатків

Після завершення конфігурації програми «Особистий кабінет» і перезавантаження веб-сервера, додаток буде доступно за адресою http://localhost:8080/PersonalCabinet/

3.2 Опис графічного інтерфейсу

Сторінка «Account information»

На даній сторінці знаходиться основна інформація з різних відділів (Billing, Services, Support) (рисунок 3.13). У секції Billing показаний поточний баланс користувача, дата до якої потрібно сплатити рахунок і опція відправлення електронних чеків. У секції Services розміщені панелі сервісів, де вказані статус сервісу, назва тарифу і посилання на доступні тарифні плани. Секція Support містить контактну інформацію підприємства (телефон, електронна адреса).

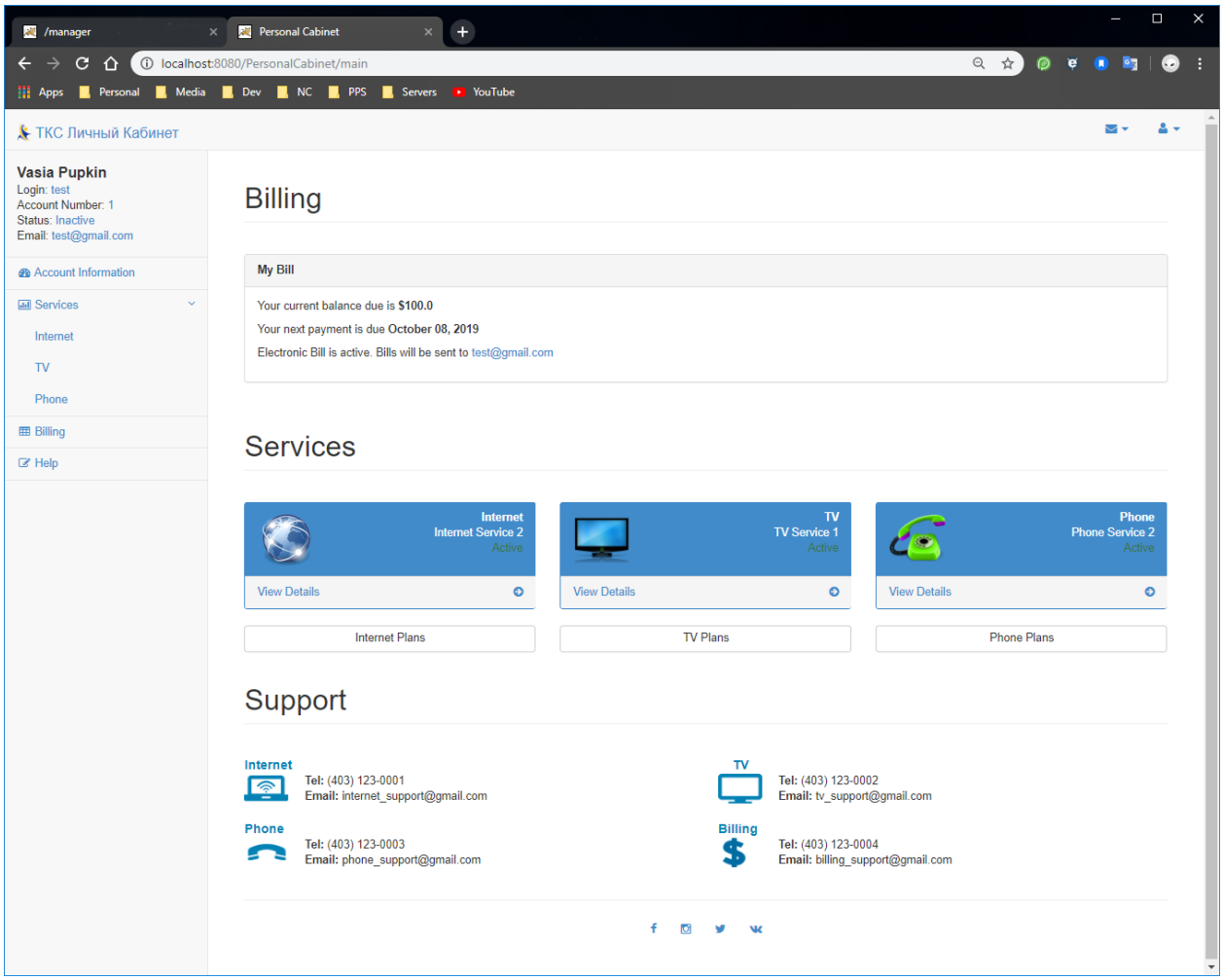


Рисунок 3.13 Сторінка «Account information»

Сторінки тарифних планів

При переході на сторінку тарифних планів для кожного із сервісів відображаються доступні тарифні плани, їх вартість і технічні характеристики (рисунок 3.14). Якщо тарифний план уже встановлено, то відповідний керуючий елемент буде недоступний. В іншому випадку користувач може застосувати обраний план.

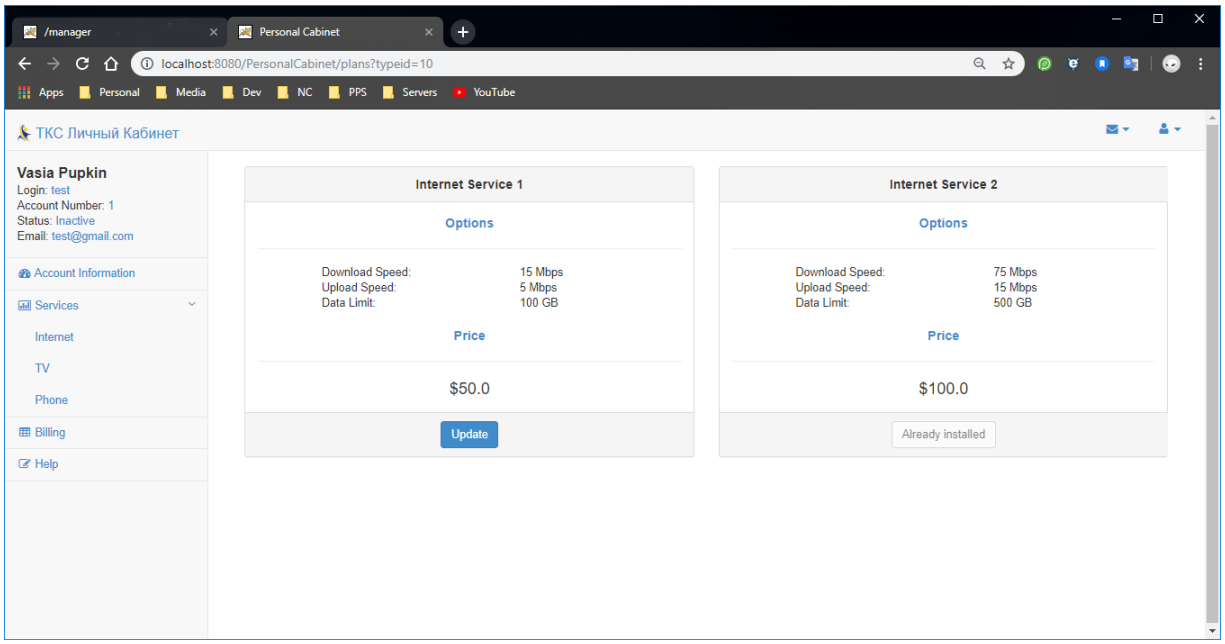


Рисунок 3.14 Тарифні плани Інтернет сервісу

Для Інтернет-сервісу присутні наступні технічні характеристики: швидкість завантаження / вивантаження і щомісячний ліміт даних.

Для ТВ-сервісу присутні наступні технічні характеристики: кількість каналів і підтримка надвисокої чіткості (рисунок 3.15).

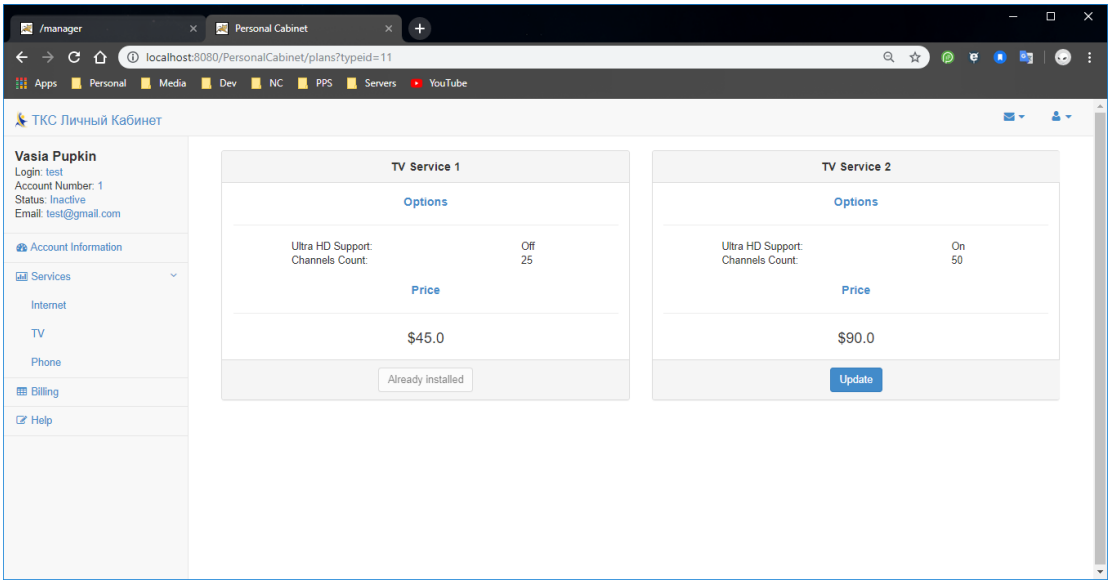


Рисунок 3.15 Тарифні плани ТВ-сервісу

Для сервісу Телефон присутні наступні технічні характеристики: наявність голосової пошти, щомісячний обсяг мобільного інтернету, кількість безкоштовних хвилин на місяць (рисунок 3.16).

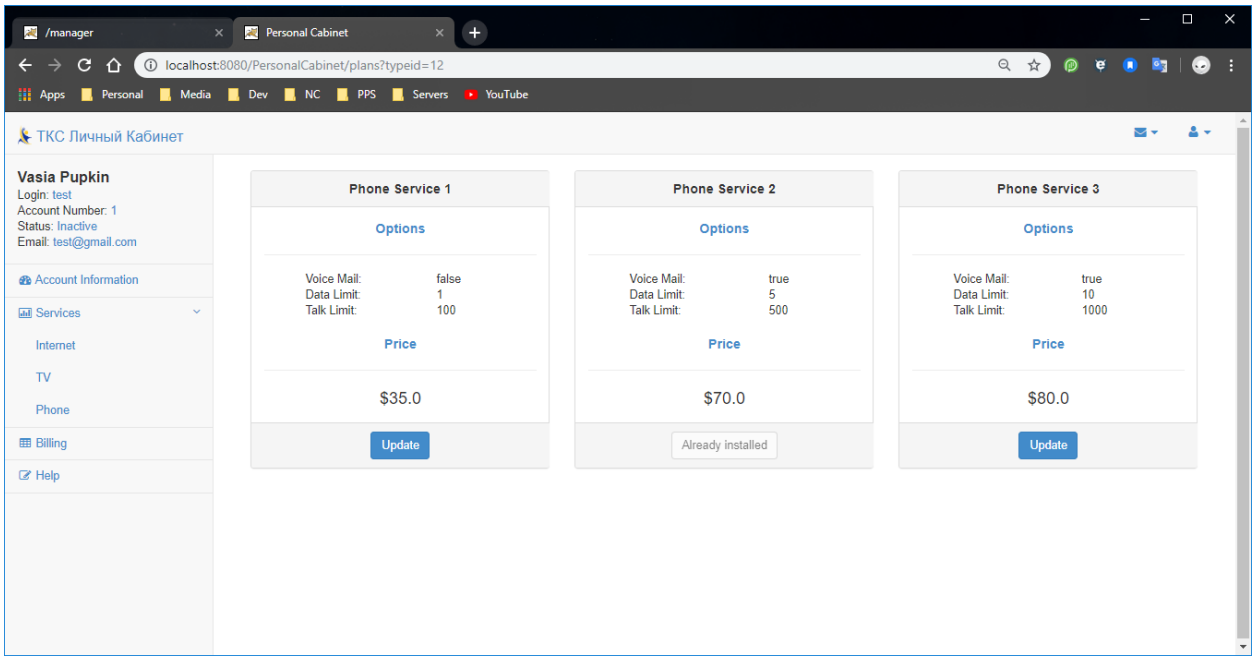


Рисунок 3.16 Тарифні плани сервісу Телефон

Сторінка Інтернет-сервісу

На сторінці Інтернет-сервісу відображаються технічні характеристики тарифного плану, встановлене обладнання (наприклад, модем), адреси електронної пошти та кількість використаного трафіку (рисунок 3.17).

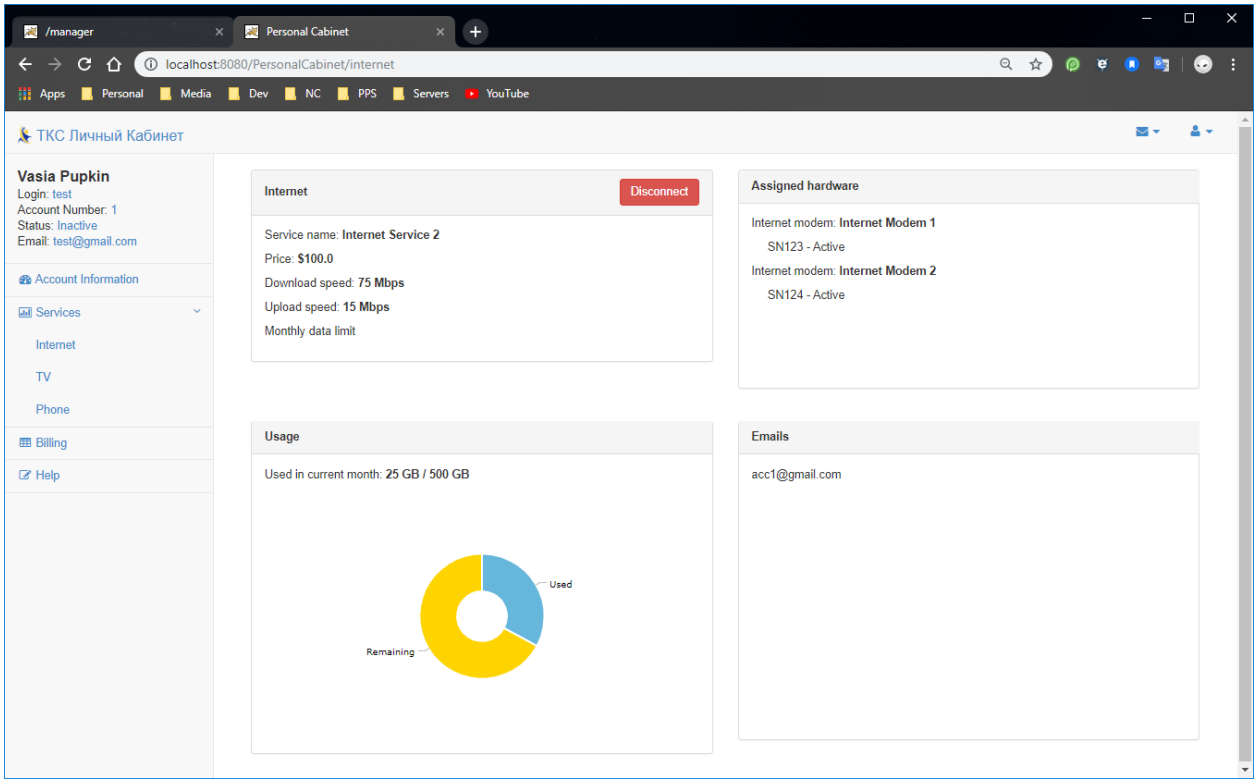


Рисунок 3.17 Сторінка Інтернет-сервісу

Сторінка ТВ-сервісу

На сторінці ТВ-сервісу відображаються технічні характеристики тарифного плану та встановлене обладнання (рисунок 3.18).

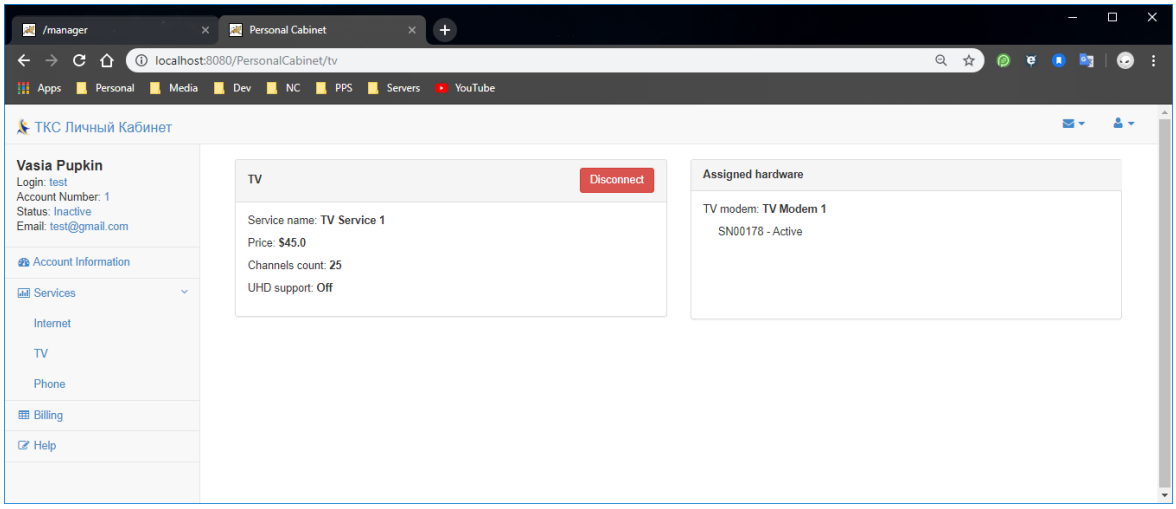


Рисунок 3.18 Сторінка ТВ-сервісу

Сторінка сервісу Телефон

На сторінці Телефон-сервісу відображаються технічні характеристики тарифного плану, встановлене обладнання (рисунок 3.19).

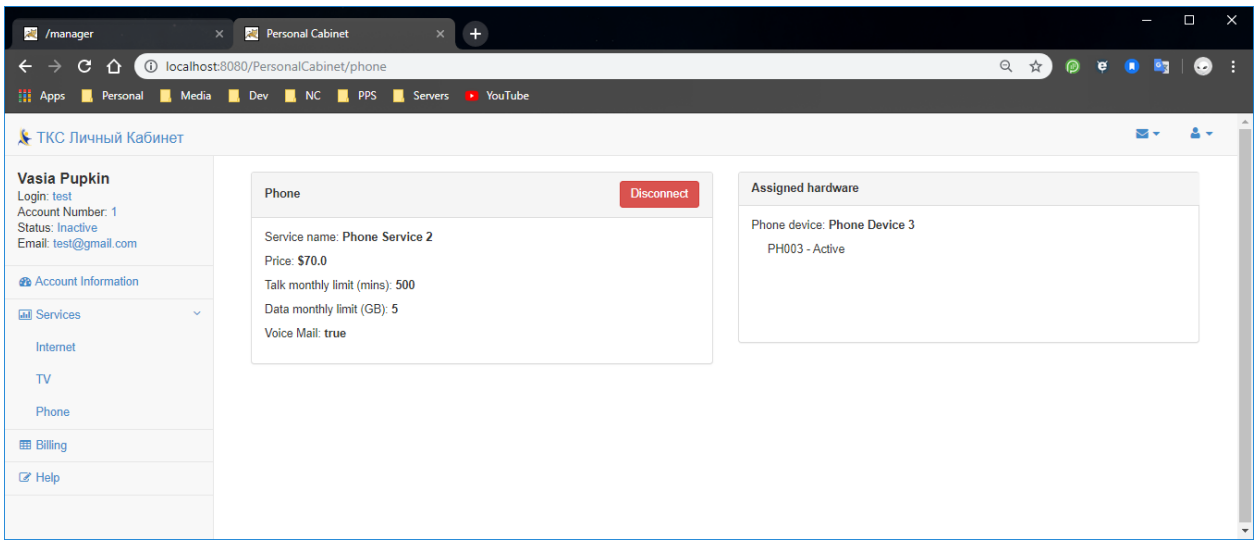


Рисунок 3.19 Сторінка сервісу Телефон

Сторінка «*Billing*»

На сторінці Billing сервісу відображається інформація про поточний баланс і історія платежів (рисунок 3.20).

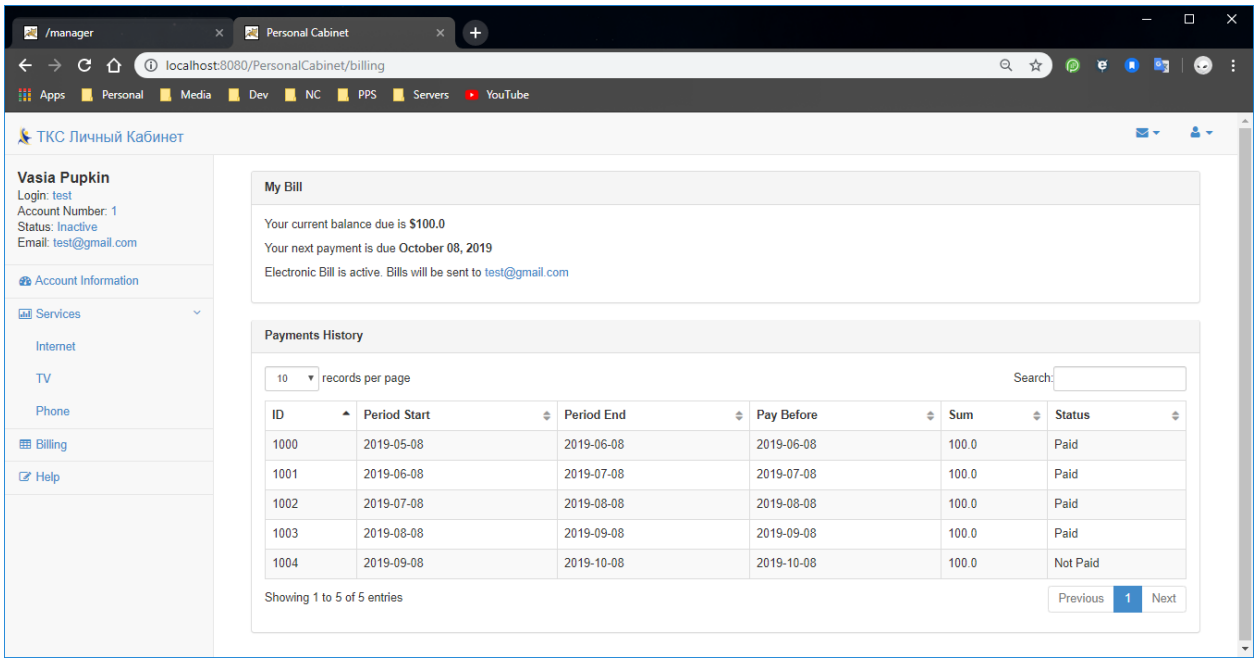


Рисунок 3.20 Сторінка з відображенням балансу

Сторінка «Help»

На сторінці Help-сервісу вказана контактна інформація про підприємство. У перспективі тут також можна розмістити посилання на онлайн чат для вирішення технічних проблем (рисунок 3.21).

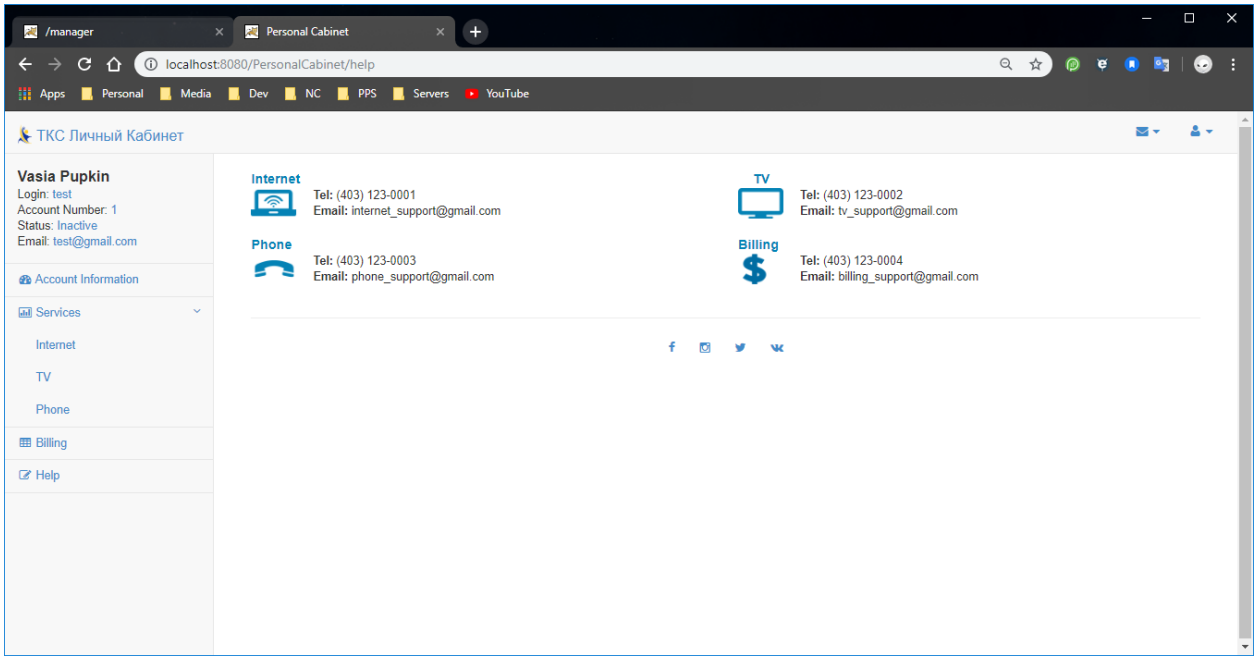


Рисунок 3.21 Сторінка з контактною інформацією технічної підтримки.

ВИСНОВКИ

В ході виконання дипломної роботи була спроектована інформаційна система підприємства ТОВ «Сумські телекомсистеми». Даний проект призначений для розширення можливостей взаємодії підприємства з клієнтами, підвищення якості обслуговування і зниження витрат на обслуговуючий персонал.

Було проведено аналіз основних параметрів і функцій, які повинні бути реалізовані в даній інформаційній системі. За допомогою діаграм потоків даних ми побудували і проілюстрували концептуальну модель системи, визначили основні процеси, потоки і сховища даних.

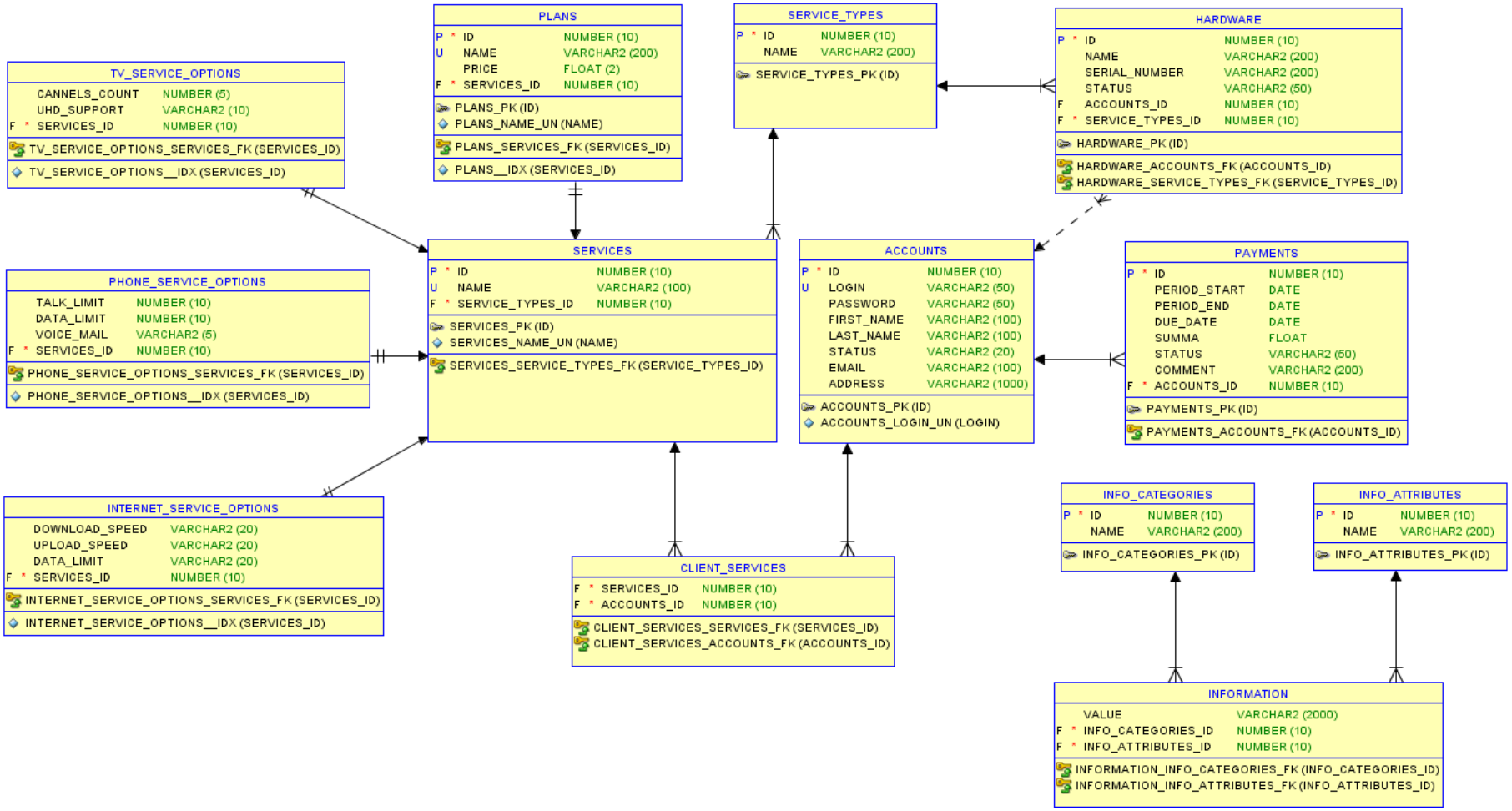
На основі даної інформаційної системи було реалізовано веб-додаток «Особистий кабінет». Після завершення розробки і тестування додатку, аналізу його ефективності, можна зробити висновок, що в цілому дана реалізація є успішною й може бути використана у подальшому для модернізації існуючої інформаційної системи підприємства «Сумські телекомсистеми».

СПИСОК ЛІТЕРАТУРИ

1. Бази даних. Класифікація баз даних. [Електронний ресурс] – Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L5.htm
2. Визначення та класифікація СУБД [Електронний ресурс] – Режим доступу: https://studopedia.com.ua/1_43735_viznachennya-ta-klasifikatsiya-subd.html
3. Вхід в систему My Vodafone [Електронний ресурс] – Режим доступу: https://my.vodafone.ua/
4. Вхід в систему самообслуговування Koodo Mobile [Електронний ресурс] – Режим доступу: https://www.koodomobile.com/
5. Вхід до особистого кабінету MyShaw [Електронний ресурс] – Режим доступу: https://www.shaw.ca/store/
6. Головна сторінка підприємства ООО «Сумські телекомсистеми» [Електронний ресурс] – Режим доступу: http://tks.sumy.ua/
7. Корнієнко, М.М. Інформатика. Бази даних. Системи управління базами даних. Microsoft Access: Теоретичні основи, приклади та завдання, практичні роботи. – Видавництво «Ранок», 2009.
8. Кузиков, Б.О. Базы данных и информационные системы. Физическое моделирование БД. [Електронний ресурс] – Режим доступу: https://dl.sumdu.edu.ua/textbooks/89707/364109/index.html
9. Курс «Проектирование информационных систем» Национального Открытого Университета «ИНТУИТ» [Електронний ресурс] - Режим доступу: http://www.intuit.ru/studies/courses/2195/55/info , вільний — Заголовок з екрану.
10. Фаулер Мартин UML Основы. Третье издание. – Символ-Плюс, 2018.

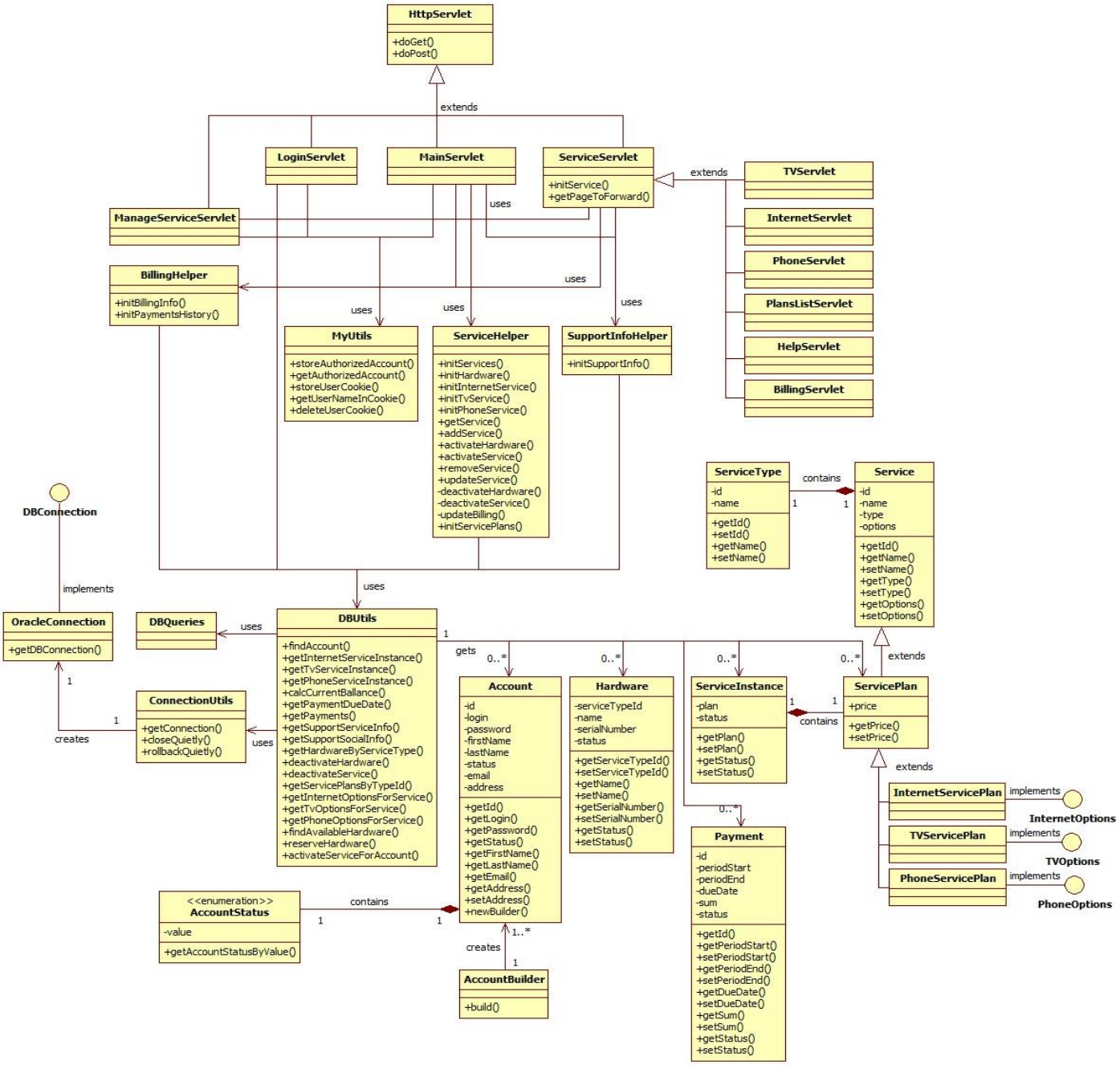
ДОДАТОК 1

ER-діаграма. Модель відносин



ДОДАТОК 2

Діаграма класів додатку «Особистий кабінет»



ДОДАТОК 3

```
CREATE TABLE accounts (
  id NUMBER(10) NOT NULL,
  login VARCHAR(50) NOT NULL,
  password VARCHAR(50),
  first_name VARCHAR(100),
  last_name VARCHAR(100),
  status VARCHAR(20) NOT NULL,
  email VARCHAR(100),
  address  VARCHAR(1000)
  , CONSTRAINT accounts_pk PRIMARY KEY (id)
  , CONSTRAINT accounts_login_uk UNIQUE (login)
  , CONSTRAINT accounts_status_fixed_values
CHECK (status IN ('Active', 'Inactive'))
);

CREATE TABLE plans (
  id NUMBER(10) NOT NULL,
  name VARCHAR(200) NOT NULL,
  price FLOAT DEFAULT 0
  , CONSTRAINT plans_pk PRIMARY KEY (id)
  , CONSTRAINT plans_name_uk UNIQUE (name)
  , CONSTRAINT plans_price_not_negative CHECK (price >= 0)
);

CREATE TABLE service_types (
  id NUMBER(10) NOT NULL,
  name VARCHAR(200) NOT NULL
  , CONSTRAINT service_types_pk PRIMARY KEY (id)
  , CONSTRAINT service_types_name_uk UNIQUE (name)
);

CREATE TABLE services (
  id NUMBER(10) NOT NULL,
  name VARCHAR(200) NOT NULL,
  type_id NUMBER(10),
  plan_id NUMBER(10)
  , CONSTRAINT services_pk PRIMARY KEY (id)
  , CONSTRAINT services_name_uk UNIQUE (name)
  , CONSTRAINT services_type_id_fk
      FOREIGN KEY (type_id)
      REFERENCES service_types (id)
  , CONSTRAINT services_plan_id_fk
      FOREIGN KEY (plan_id)
      REFERENCES plans (id)
);

CREATE TABLE client_services (
  account_id NUMBER(10),
  service_id NUMBER(10)
  , CONSTRAINT client_services_uk UNIQUE (account_id, service_id)
  , CONSTRAINT client_services_acc_id_fk
      FOREIGN KEY (account_id)
      REFERENCES accounts (id)
     ON DELETE CASCADE
  , CONSTRAINT client_services_serv_id_fk
      FOREIGN KEY (service_id)
      REFERENCES services (id)
```

```
    ON DELETE CASCADE
);


CREATE TABLE payments (
  id NUMBER(10) NOT NULL,
  account_id NUMBER(10) NOT NULL,
  period_start DATE NOT NULL,
  period_end DATE NOT NULL,
  due_date DATE NOT NULL,
  summa NUMBER(10) DEFAULT 0,
  status VARCHAR2(50) DEFAULT 'Not Paid',
  "COMMENT" VARCHAR2(200)
  , CONSTRAINT payments_pk PRIMARY KEY (id)
  , CONSTRAINT payments_account_id_fk
      FOREIGN KEY (account_id)
      REFERENCES accounts (id)
  , CONSTRAINT payments_status_values_chk
      CHECK (status IN ('Paid', 'Not Paid'))
  , CONSTRAINT payments_period_chk
      CHECK (period_start <= period_end)
);

CREATE TABLE hardware (
  id NUMBER(10) PRIMARY KEY NOT NULL,
  account_id NUMBER(10) REFERENCES accounts (id),
  service_type_id NUMBER(10) REFERENCES service_types (id),
  name VARCHAR2(200) NOT NULL,
  serial_number VARCHAR2(200) NOT NULL,
  status VARCHAR2(50) DEFAULT 'Inactive'
  , CONSTRAINT hw_status_values_chk
      CHECK (status in ('Active', 'Inactive'))
);

CREATE TABLE info_categories (
  id NUMBER(10) NOT NULL,
  name VARCHAR(200) NOT NULL
  , CONSTRAINT info_categories_pk PRIMARY KEY (id)
  , CONSTRAINT info_categories_name_uk UNIQUE (name)
);
CREATE TABLE info_attributes (
  id NUMBER(10) NOT NULL,
  name VARCHAR(200) NOT NULL
  , CONSTRAINT info_attributes_pk PRIMARY KEY (id)
  , CONSTRAINT info_attributes_uk UNIQUE (name)
);

CREATE TABLE information (
  attr_id NUMBER(10) NOT NULL,
  category_id NUMBER(10) NOT NULL,
  value VARCHAR2(500)
  , CONSTRAINT information_attr_id_fk
      FOREIGN KEY (attr_id)
      REFERENCES info_attributes (id)
  , CONSTRAINT information_cat_id_fk
      FOREIGN KEY (category_id)
      REFERENCES info_categories (id)
);

CREATE TABLE internet_service_options (
```

```
  service_id NUMBER(10) NOT NULL,
  download_speed VARCHAR(20) NOT NULL,
  upload_speed VARCHAR(20) NOT NULL,
  data_limit VARCHAR(20) NOT NULL
  , CONSTRAINT int_serv_opt_fk
      FOREIGN KEY (service_id)
      REFERENCES services (id)
);

CREATE TABLE tv_service_options (
  service_id NUMBER(10) NOT NULL,
  channels_count NUMBER(5) NOT NULL,
  uhd_support VARCHAR(10) NOT NULL
  , CONSTRAINT tv_serv_opt_fk
      FOREIGN KEY (service_id)
      REFERENCES services (id)
  , CONSTRAINT tv_serv_opt_uhd_values
      CHECK (uhd_support IN ('On', 'Off'))
);

CREATE TABLE phone_service_options (
  service_id NUMBER(10) NOT NULL,
  talk_limit NUMBER(10) NOT NULL,
  data_limit NUMBER(10) NOT NULL,
  voice_mail VARCHAR2(5) DEFAULT 'Off'
  , CONSTRAINT ph_serv_opt_fk
      FOREIGN KEY (service_id)
      REFERENCES services (id)
  , CONSTRAINT ph_serv_opt_vm_values
      CHECK (voice_mail IN ('On', 'Off'))
);
```

ДОДАТОК 4

```
CREATE SEQUENCE accounts_seq
  START WITH 1
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER accounts_insert_trigger
  BEFORE INSERT ON accounts
  FOR EACH ROW
BEGIN
  SELECT accounts_seq.nextval
    INTO :new.id
    FROM dual;
END;
/

CREATE SEQUENCE service_types_seq
  START WITH 10
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER service_types_insert_trigger
  BEFORE INSERT ON service_types
  FOR EACH ROW
BEGIN
  SELECT service_types_seq.nextval
    INTO :new.id
    FROM dual;
END;
/

CREATE SEQUENCE services_seq
  START WITH 100
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER services_insert_trigger
  BEFORE INSERT ON services
  FOR EACH ROW
BEGIN
  SELECT services_seq.nextval
    INTO :new.id
    FROM dual;
END;
/
CREATE SEQUENCE payments_seq
  START WITH 1000
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER payments_insert_trigger
  BEFORE INSERT ON payments
  FOR EACH ROW
BEGIN
```

```
  SELECT payments_seq.nextval
    INTO :new.id
    FROM dual;
END;
/

CREATE SEQUENCE hardware_seq
  START WITH 10000
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER hardware_insert_trigger
  BEFORE INSERT ON hardware
  FOR EACH ROW
BEGIN
  SELECT hardware_seq.nextval
    INTO :new.id
    FROM dual;
END;
/

CREATE SEQUENCE plans_seq
  START WITH 20000
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER plans_insert_trigger
  BEFORE INSERT ON plans
  FOR EACH ROW
BEGIN
  SELECT plans_seq.nextval
    INTO :new.id
    FROM dual;
END;
/

CREATE SEQUENCE info_cat_seq
  START WITH 1
  INCREMENT BY 1
  CACHE 100;
/

CREATE OR REPLACE TRIGGER info_cat_insert_trigger
  BEFORE INSERT ON info_categories
  FOR EACH ROW
BEGIN
  SELECT info_cat_seq.nextval
    INTO :new.id
    FROM dual;
END;
/

CREATE SEQUENCE info_attr_seq
  START WITH 10
  INCREMENT BY 1
  CACHE 100;
/
```

```
CREATE OR REPLACE TRIGGER info_attr_insert_trigger
  BEFORE INSERT ON info_attributes
  FOR EACH ROW

BEGIN
  SELECT info_attr_seq.nextval
    INTO :new.id
    FROM dual;
END;
/
```