МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
червня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційний веб-портал станції технічного обслуговування "Автограф"»

здобувача групи ІН-92 Харути Артема Олексійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Харута А.О.
(підпис)

Керівник,
старший викладач, кандидат фізико-математичних наук Олексієнко Г.А. _____
(підпис)

Суми – 2023

Сумський державний університет

Центр заочної, дистанційної та вечірньої форм навчання

Кафедра комп'ютерних наук

«Затверджую»

В.о. завідувача кафедри

Ігор ШЕЛЕХОВ

_____

(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»

здобувача групи ІН – 92 Харута Артем Олексійович

1. Тема роботи: « Інформаційний веб-портал станції технічного обслуговування "Автограф »

затверджую наказом по СумДУ від

2. Термін здачі здобувачем кваліфікаційної роботи

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1)Виконати загальний інформаційний огляд основних методів реалізації завдання. 2)Дослідити існуючі аналоги інформаційних систем. 3)Обрати методи реалізації та програмне забезпечення. 4)Виконати програмну реалізацію

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис)

Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	Оформлення пропуску на місце практики	14.04.2023-15.04.2023	
2	Огляд літератури по темі практики	16.04-20.04.2023	
3	Дослідити існуючі аналоги інформаційних систем	20.04-21.04.2023	
4	Вибір програмних засобів рішення задачі	21.04-30.04.2023	
5	Виконати програмну реалізацію	30.04-16.05.2023	
6	Підготовка до захисту звіту	16.05.23-17.05.2023	

Здобувач вищої освіти _____ (підпис)

Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 50 стр., 11 рис,. 19 використаних джерел., 1 додаток.

Обґрунтування актуальності теми роботи – Розробка інформаційного веб-порталу для сайту СТО "Автограф"" є надзвичайно актуальною та має практичне значення. У сучасну епоху діджиталізації та розвитку інтернет-технологій, веб-портали стають невід'ємною частиною багатьох сфер бізнесу, в тому числі і автомобільної індустрії.

Об’єкт дослідження — Інформаційний веб-портал станції технічного обслуговування “Автограф”

Мета роботи — проектування та розробка інформаційного веб-порталу станції технічного обслуговування “Автограф”

Методи дослідження — системно-інформаційний аналіз, інформаційне моделювання та комп’ютерний експеримент.

Результати — проаналізовано літературу та розглянуто поняття інформаційних систем, їх класифікацію та переваги їх використання. Проаналізовано існуючі інформаційні системи та підходи до їх реалізації. Розглянувши та проаналізувавши існуючі рішення, розроблено власний алгоритм для вирішення поставленої задачі. З різних доступних інструментів та підходів обрано та обґрунтовано СУБД, мову програмування, фреймворк та середовище розробки. Розроблено базу даних та реалізовано додаток у вигляді веб-сайту. Додаток виконує поставлене завдання і дозволяє користувачам зручно взаємодіяти з інформаційною системою обліку клієнтів на СТО "Автограф".

ІНОФРМАЦІЙНІ СИСТЕМИ, SPRING FRAMEWORK, JAVA, MYSQL.

ЗМІСТ

ВСТУП

У своєму звіті про бакалаврську роботу я хотів би розповісти про розробку інформаційної системи для веб-сайту станції технічного обслуговування "Автограф". Метою цієї розробки є створення початкового бекенду, що включає системи аутентифікації, реєстрації та бронювання.

У сучасному світі веб-сайти стали важливим інструментом підтримки бізнесу та забезпечення ефективної взаємодії з клієнтами. Станція технічного обслуговування "Автограф" також вирішила розширити свою присутність в Інтернеті, створивши веб-сайт, який дозволив би клієнтам зручно та швидко отримати доступ до її послуг.

Однак успішний веб-сайт вимагає добре організованої інформаційної системи. В рамках моєї переддипломної практики я розробив бекенд веб-сайту СТО "Автограф", який включає в себе такі функції, як системи автентифікації, реєстрації та бронювання.

Аутентифікація є важливим компонентом будь-якої інформаційної системи, оскільки вона забезпечує безпеку даних та контрольований доступ до функцій веб-сайту. Впровадження системи автентифікації передбачає ідентифікацію користувачів, перевірку їхніх облікових записів і надання відповідних прав доступу до функцій веб-сайту.

Реєстрація дозволяє новим користувачам створити обліковий запис на сайті СТО "Автограф". Це дає можливість зберігати персональні дані клієнта та надавати доступ до індивідуальних налаштувань і вподобань.

Нарешті, компонентом системи, що розробляється, є система бронювання. Ця функція дозволить клієнтам бронювати технічне обслуговування та ремонт автомобілів через сайт "Автографа". Це спростить процес бронювання та дозволить клієнтам обирати зручну дату та час.

У цьому звіті детально описано процес розробки та впровадження початкового бек-енду інформаційної системи для веб-сайту "Автограф". Він описує використані технології, методологію тестування та ключові етапи процесу розробки. Також представлено результати роботи, аналіз отриманих даних та висновки щодо ефективності розробленої системи.

Метою цього звіту є надання всебічного огляду розробки інформаційної системи для веб-сайту Центру надання послуг підпису з функціоналом системи авторизації, реєстрації та бронювання. Звіт має на меті підкреслити важливість веб-сайту та інформаційної системи для покращення взаємодії з клієнтами та забезпечення їхнього комфорту і задоволеності.

1.ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Загальна інформація про основні методи реалізації взаємодії з клієнтом на сайті СТО

Загальна інформація про основні способи взаємодії з клієнтами на сайтах СТО може включати наступні варіанти.

Статичний контент є основним способом взаємодії з клієнтами на багатьох сайтах. У цьому випадку інформація на сайті є статичною і не змінюється залежно від користувача або його поведінки. На сайті СТО статичний контент може включати загальну інформацію про сервіс, контактні дані, графік роботи, інформацію про команду і професійний досвід механіка.

Форми зворотного зв'язку - це поширений спосіб, за допомогою якого клієнти можуть надсилати свої запити або запитання безпосередньо з веб-сайту. Це може бути контактна форма, де користувачі можуть ввести своє ім'я, електронну пошту та повідомлення. Цей метод дозволяє зручно збирати та обробляти запити клієнтів.

Онлайн-чат і чат-боти можна використовувати для покращення взаємодії з клієнтами. За допомогою чату клієнти можуть ставити запитання, звертатися за порадою, отримувати необхідну інформацію про послуги та бронювати зустрічі. Це може бути змодельовано за допомогою реальної людини або автоматизованого чат-бота.

Функції реєстрації та автентифікації. Система реєстрації та автентифікації дозволяє клієнтам створювати обліковий запис і входити на сайт. Це дозволяє їм зберігати персоналізовану інформацію, таку як історія замовлень, улюблені послуги та особисті уподобання.

Система призначень. Система попереднього запису дозволяє клієнтам обирати зручну дату та час для проведення технічного обслуговування. Клієнт

може переглянути доступні слоти, вибрати необхідний слот, ввести відповідну інформацію та підтвердити запис.

Інтеграція з платіжними системами. При необхідності можлива інтеграція з платіжними системами для прийому оплати за послуги та депозитів. Можлива оплата онлайн кредитною карткою, електронним гаманцем або іншими способами.

Загальна інформація про основні способи реалізації взаємодії з клієнтами на сайтах СТО, а також інші варіанти, які можуть бути корисними:

Інтеграція з соціальними мережами. Додавання можливості входу або реєстрації через акаунти соціальних мереж (наприклад, Facebook, Google) дозволяє клієнтам легко входити на сайт без необхідності створювати новий обліковий запис. Це спрощує процес аутентифікації та робить його більш зручним для користувачів.

Онлайн-консультації та чат підтримки. Завдяки онлайн-консультаціям та чату підтримки клієнти можуть негайно отримати відповіді на свої запитання або поспілкуватися з представником СТО. Це створює новий рівень комунікації та забезпечує клієнтам більш швидку підтримку.

Калькулятор вартості послуг. Додавання калькулятора вартості послуг на сайт дозволяє клієнтам швидко оцінити очікувану вартість технічного обслуговування і ремонту. Користувачі можуть вказати необхідну послугу або ремонт і отримати оцінку, уникаючи непорозумінь.

Відгуки та оцінки клієнтів. Додавання розділу для відгуків та оцінок клієнтів може показати довіру та репутацію станції технічного обслуговування. Клієнти можуть залишати свої враження та оцінки після отримання послуги, що може допомогти іншим користувачам зробити вибір і дати їм більше спокою.

1.2. Огляд існуючих інформаційних систем

Першим прикладом буде сайт автосервісу «AIC Суми» https://sto.sumy.ais.ua/ . На цьому сайті реалізовано такий функціонал :

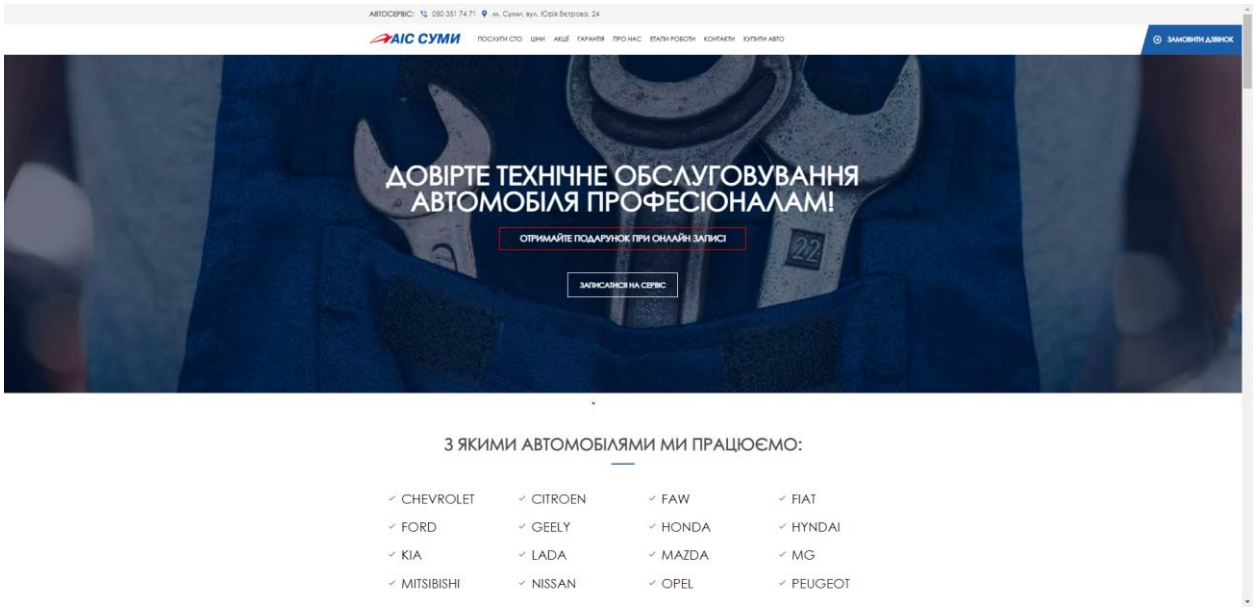

Рисунок 1.1 – Головна сторінка «AIC Суми»

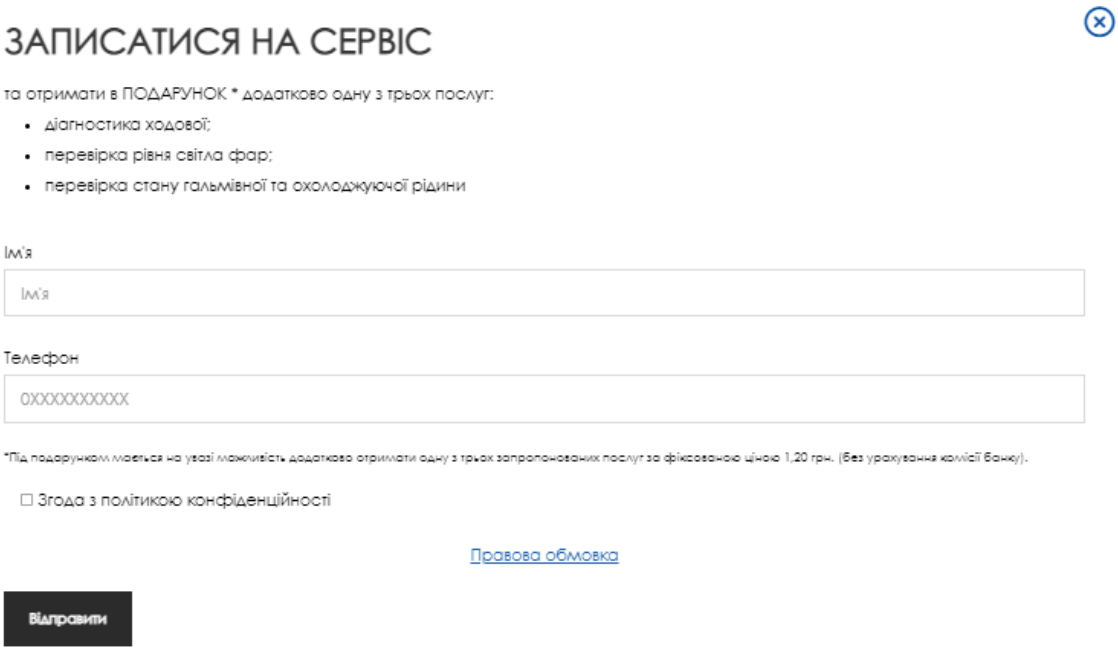

Рисунок 1.2 – Форма бронювання

Онлайн запис: Де клієнт може вказати в полях форми своє ім’я та номер телефону, та надіслати їх адміністратору.

Другим прикладом функціоналу для сайту Автосервісу буде: автосервіс «CARPRIDE» https://bmw-pride.com.ua/ua/ , на цьому сайті реалізовані більш масштабні інформаційні системи, такі як:

1. Кошик і можливість бронювання товарів
2. Пошук по позиціях
3. Можливість залишити фідбек
4. Також інтегровано взаємодія з соціальними мережами Facebook і Twitter

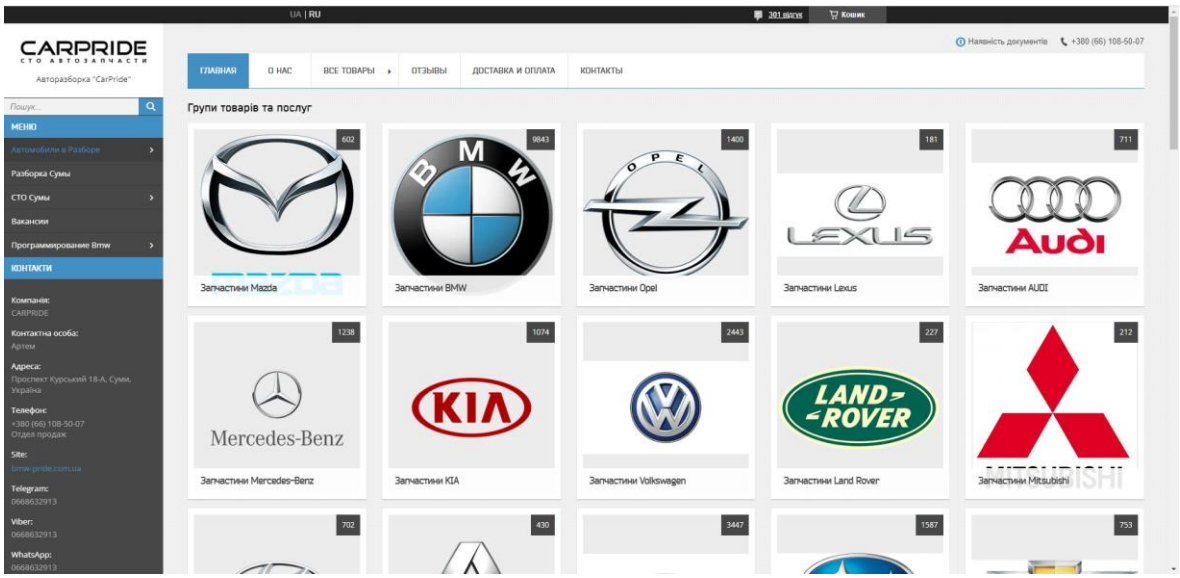

Рисунок 1.3 – Головна сторінка «CARPRIDE»

Наступний приклад «Garant» https://garant.sumy.ua/ , на цьому сайті також реалізована система запису на прийом, але з деякими відмінностями, наприклад з можливістю обрати потрібну послугу та вказати марку автомобіля

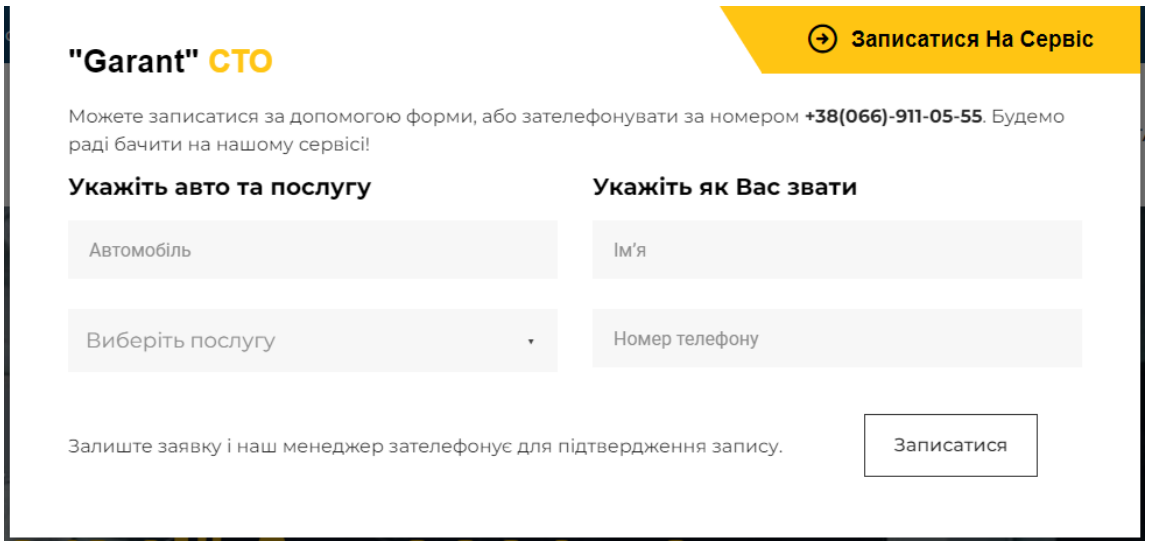

Рисунок 1.4 – Форма бронювання «Garant»

1.3 Постановка задачі

Метою проекту є розробка системи реєстрації/автентифікації та бронювання для інформаційної системи на сайті СТО "Автограф". Система повинна забезпечити клієнтам легкий доступ до СТО та спростити процес бронювання. Основні вимоги до функціональності системи реєстрації/авторизації та бронювання:

Реєстрація користувачів:

Користувачі повинні мати можливість створити обліковий запис на сайті. Для цього вони повинні ввести необхідну особисту інформацію, таку як ім'я, прізвище, електронну пошту та пароль. Процес реєстрації повинен бути простим та інтуїтивно зрозумілим для клієнта.

Авторизація користувачів:

Зареєстровані користувачі повинні мати можливість увійти на сайт. Це забезпечить безпеку та конфіденційність ваших персональних даних. Це можна зробити, ввівши адресу електронної пошти та пароль або використовуючи свій обліковий запис у соціальних мережах.

Захист даних користувача.

Система повинна забезпечувати безпеку і конфіденційність персональних даних користувачів. Паролі повинні бути зашифровані і зберігатися в базі даних, а доступ до персональних даних повинен бути обмежений тільки для авторизованих користувачів.

Системи бронювання прийому

Система повинна дозволяти клієнтам обирати зручну для них дату та час для технічного обслуговування та записуватись на прийом. Вони повинні мати можливість переглянути доступні слоти, вибрати потрібний слот і ввести необхідну інформацію, таку як дані про транспортний засіб і тип послуги.

Сповіщення та нагадування

Система повинна мати можливість надсилати клієнтам сповіщення та нагадування про їхні бронювання. Сюди входять повідомлення, що надсилаються на електронну пошту або мобільний пристрій перед зустріччю, а також зміни в розкладі обслуговування.

Доступ адміністратора:

Система повинна мати окремий розділ для адміністраторів для перегляду, управління та зміни даних клієнтів, графіків обслуговування та доступних послуг. Адміністратори повинні мати можливість додавати та видаляти зареєстрованих користувачів, а також вносити зміни в інші частини системи.

Ці основні вимоги до функціональності системи реєстрації, автентифікації та бронювання слід враховувати при розробці веб-сайту та реалізації цих функцій. Вони покликані зробити взаємодію між СТО "Автограф" та клієнтами комфортною і безперешкодною.

2.ВИБІР МЕТОДІВ РЕАЛІЗАЦІЇ ЗАВДАННЯ

2.1.Вибір мови програмування

PHP [1]

Переваги:

Широко розповсюджена та популярна: PHP - одна з найпопулярніших мов програмування для розробки веб-додатків. Вона має велику спільноту розробників, і на ній легко знайти допомогу та ресурси.

Численні фреймворки: PHP має велику кількість фреймворків, таких як Laravel, Symfony і CodeIgniter, які полегшують розробку веб-додатків і включають багато вбудованих функцій.

Хороша підтримка баз даних: PHP має розширення для роботи з різними базами даних, включаючи MySQL, для задоволення потреб систем реєстрації, автентифікації та бронювання.

Недоліки:

Масштабованість: PHP - не найкращий вибір для великих, складних проектів через його обмежену масштабованість у порівнянні з іншими мовами програмування.

Безпека: Старі версії PHP мали проблеми з безпекою, які були вирішені в останні роки. Проте, при розробці PHP-додатків важливо дотримуватися найкращих практик безпеки.

Python [2]

Переваги:

Зручний та елегантний синтаксис: Python відомий своїм простим і зрозумілим синтаксисом, який полегшує розробку та розуміння коду.

Багата екосистема: Python має низку сторонніх бібліотек та фреймворків, таких як Django та Flask, які прискорюють розробку веб-додатків.

Потужна підтримка наукових обчислень: Python популярна у спільноті наукових обчислень і є корисною, коли потрібен додатковий функціонал для аналізу даних та статистики.

Недоліки:

Продуктивність: Python може бути менш ефективною, ніж інші мови, такі як PHP та Java, особливо при обробці великих обсягів даних.

Великі вимоги до пам'яті: Python використовує більше пам'яті, що може бути проблемою, коли ресурси сервера обмежені.

Ruby [3]

Переваги:

Елегантний та зрозумілий синтаксис: Ruby має гарний синтаксис, який дозволяє розробникам швидко писати та читати код.

Фреймворк Rails: Ruby on Rails - один з найпопулярніших фреймворків для веб-розробки, який надає багату функціональність і прискорює розробку веб-додатків.

Він базується на принципах "юзабіліті" та "конвенції важливіші за конфігурацію": Ruby забезпечує стандартизований підхід до розробки, полегшуючи спільну роботу розробників та забезпечуючи узгодженість коду.

Недоліки:

Продуктивність: Ruby може бути повільною порівняно з іншими мовами програмування, що може бути проблемою при обробці великих обсягів даних або коли потрібна висока продуктивність.

Брак популярності: Рубі має меншу спільноту розробників, ніж PHP та Python, що може вплинути на доступність допомоги та ресурсів.

Java [4]

Переваги:

Кросплатформеність: Java працює на різних операційних системах, що спрощує розгортання і масштабування веб-додатків.

Велика екосистема: Java має багатий набір фреймворків і бібліотек, як-от Spring і Hibernate, які допомагають розробляти високоякісні, масштабовані додатки.

Висока продуктивність: Java відома своєю високою продуктивністю і чуйністю, що робить її гарним вибором для веб-додатків з інтенсивним використанням даних і високим навантаженням.

Недоліки:

Складність: Java є більш формальною і складною мовою порівняно з іншими мовами, такими як PHP і Python, вимагаючи більше коду для досягнення тих самих результатів.

Високі вимоги до ресурсів: Java може вимагати більше пам'яті та обчислювальних ресурсів, що може бути проблемою в умовах обмежених серверних середовищ.

Після ретельного розгляду різних варіантів мов програмування я вирішив зупинити свій вибір на користь Java для реалізації проекту сервісної станції "Автограф". Нижче наведено деякі аргументи на користь мого рішення.

Широке поширення та велика спільнота: Java - одна з найпопулярніших мов програмування у світі. Вона має велику спільноту розробників та широкий спектр бібліотек і фреймворків. Це означає, що

набагато легше знайти документацію, підручники та підтримку спільноти, що допомагає швидкому розвитку та вирішенню можливих проблем.

Кросплатформеність: Java - кросплатформна мова програмування. Це означає, що додатки, написані на Java, можуть без проблем працювати на різних операційних системах. Це важливо для проєктів, які мають використовуватися на різних пристроях і платформах.

Масштабованість: Java має потужні інструменти для розробки великих, складних проектів. Вона підтримує об'єктно-орієнтоване програмування, яке дає змогу організувати код у логічні блоки, що полегшує його розширення та супровід у майбутньому. Java також надає потужні інструменти для управління пам'яттю та оптимізації продуктивності, що важливо для веб-проектів зі зростаючим робочим навантаженням.

Багата екосистема: Java має безліч фреймворків і бібліотек для розроблення веб-застосунків; такі фреймворки, як Spring і JavaServer Faces (JSF), надають потужні інструменти для реалізації різних функцій, як-от управління базами даних, автентифікація та авторизація користувачів тощо. управління базами даних, автентифікація та авторизація користувачів тощо. Крім того, доступна велика кількість бібліотек, що дає змогу легко реалізувати рішення для різних потреб проєкту.

З огляду на ці фактори, я вирішив реалізувати систему СТО "Автограф" з використанням мови програмування Java, тому що Java є масштабованою, кросплатформеною, сумісною, доступні необхідні інструменти та ресурси, що забезпечує міцне підґрунтя для розробки.

2.2 Вибір фреймфорку [5]

Обираючи фреймворк для реалізації системи реєстрації, автентифікації та бронювання для сайту СТО "Автограф", я зупинився на Spring Security. Нижче наведені аргументи, які підтверджують моє рішення:

Масштабованість та надійність Spring Security - це потужний фреймворк для захисту веб-додатків. Він пропонує розширений захист і широкий спектр функцій для управління елементами безпеки, такими як авторизація, автентифікація, запобігання вторгненням і контроль доступу. Spring Security має репутацію масштабованості та надійності, що особливо важливо для систем, які повинні обробляти конфіденційну інформацію користувачів.

Інтеграція з екосистемою Spring Spring Security ідеально підходить для інтеграції з іншими компонентами Spring Framework: Spring Boot глибоко інтегрований з Spring MVC, що дозволяє легко встановлювати контроль доступу до різних URL-адрес і ресурсів у веб-додатку. Крім того, Spring Boot інтегрується з іншими модулями Spring, такими як Spring Data та Spring Cloud, що дозволяє зручно розробляти та розширювати вашу систему.

Розширюваність та спільнота Spring Security пропонує широкий спектр можливостей кастомізації та розширення, щоб задовольнити ваші унікальні потреби в безпеці. Ви можете легко налаштувати різні стратегії автентифікації, створити власні фільтри безпеки та розробити власні реалізації інтерфейсів. Крім того, Spring Security має активну спільноту розробників, де ви можете знайти безліч ресурсів, навчальних посібників та підтримку для вирішення потенційних проблем і викликів.

Загалом, обравши Spring Security для мого проекту Autograph Service Station, я отримав потужну та надійну систему безпеки, яку можна налаштовувати та інтегрувати з іншими компонентами Spring. Вона

захищає конфіденційну інформацію користувачів і забезпечує зручний та безпечний доступ до функцій сайту.

2.3 Середовище розробки [6]

Для розробки системи реєстрації, аутентифікації та бронювання для сайту СТО "Автограф" було обрано середовище розробки IntelliJ IDEA з наступних причин.

Потужність і функціональність: IntelliJ IDEA - це потужне та багатофункціональне інтегроване середовище розробки, спеціально розроблене для роботи з Java, Kotlin та іншими мовами програмування. Воно пропонує багато корисних функцій, таких як завершення коду, рефакторинг, налагодження та аналіз коду, що робить процес розробки простішим та продуктивнішим.

Підтримка Java та Spring: IntelliJ IDEA має вбудовану підтримку мови програмування Java та фреймворку Spring, що робить її ідеальним вибором для розробки веб-додатків на основі Java. Для розробки, налагодження і тестування Java-коду використовуються інструменти, а також функції, спеціально розроблені для підтримки розробки з використанням фреймворку Spring.

Інтеграція з іншими інструментами IntelliJ IDEA підтримує інтеграцію з широким спектром інструментів і технологій, включаючи системи контролю версій (наприклад, Git) і інструменти для написання проектів (наприклад, Gradle і Maven). Це дозволяє комфортно працювати з різними компонентами проекту та покращує співпрацю всередині вашої команди розробників.

Підтримка інших мов програмування IntelliJ IDEA підтримує інші популярні мови програмування, такі як JavaScript, HTML, CSS та Python.

Це дозволяє розробляти багатомовні додатки і використовувати різні технології в одному проекті без необхідності використання декількох редакторів і середовищ розробки.

Активна спільнота користувачів IntelliJ IDEA має велику спільноту активних користувачів і розробників, які надають можливість отримати підтримку, знайти відповіді на питання і поділитися досвідом розробки. Це особливо корисно, коли у вас виникають проблеми або вам потрібна додаткова інформація про функціональність та продуктивність IntelliJ IDEA.

2.4 Вибір бази даних [7]

Обираючи базу даних для мого Java-проекту, я зупинився на MySQL з кількох причин.

По-перше, MySQL є дуже популярною і широко використовуваною реляційною базою даних; MySQL має велику спільноту користувачів і розробників, що дозволяє легко знайти підтримку, документацію та навчальні посібники, необхідні під час розробки проекту.

По-друге, MySQL має високу продуктивність та ефективність. Вона добре обробляє великі обсяги даних і швидко виконує SQL-запити. Це особливо важливо для моїх проектів, які мають справу з великими обсягами даних.

MySQL також відома своєю надійністю та стабільністю. Я переконався, що дані мого проекту можна надійно зберігати і що він має механізм резервного копіювання для забезпечення безпеки даних. MySQL також пропонує широкий спектр функцій, необхідних для ефективної роботи з базами даних. Вона підтримує реляційні зв'язки, транзакції,

індекси, збережені процедури та багато іншого. Це дозволяє мені належним чином організовувати та керувати даними мого проекту.

Крім того, MySQL забезпечує чудову підтримку мови програмування Java, яку я використовую у своїх проектах - MySQL має драйвер JDBC, який дозволяє мені взаємодіяти з базою даних за допомогою Java. Це дозволяє дуже легко маніпулювати базою даних та інтегрувати її з кодом на Java.

З цих причин вибір MySQL для мого Java-проекту був мудрим рішенням: вона надійна, продуктивна, має широкий спектр можливостей і добре підтримує Java. Це дозволяє мені ефективно працювати з даними і робить мій проект успішним.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

5 Проектування БД з ERD

За допомогою MySQL Workbench було розроблено структуру бази даних для реалізації інформаційного порталу веб-сайту СТО "Автограф". Наведена нижче діаграма "сутність-зв'язок" (ERD) показує структуру бази даних та зв'язки між таблицями.

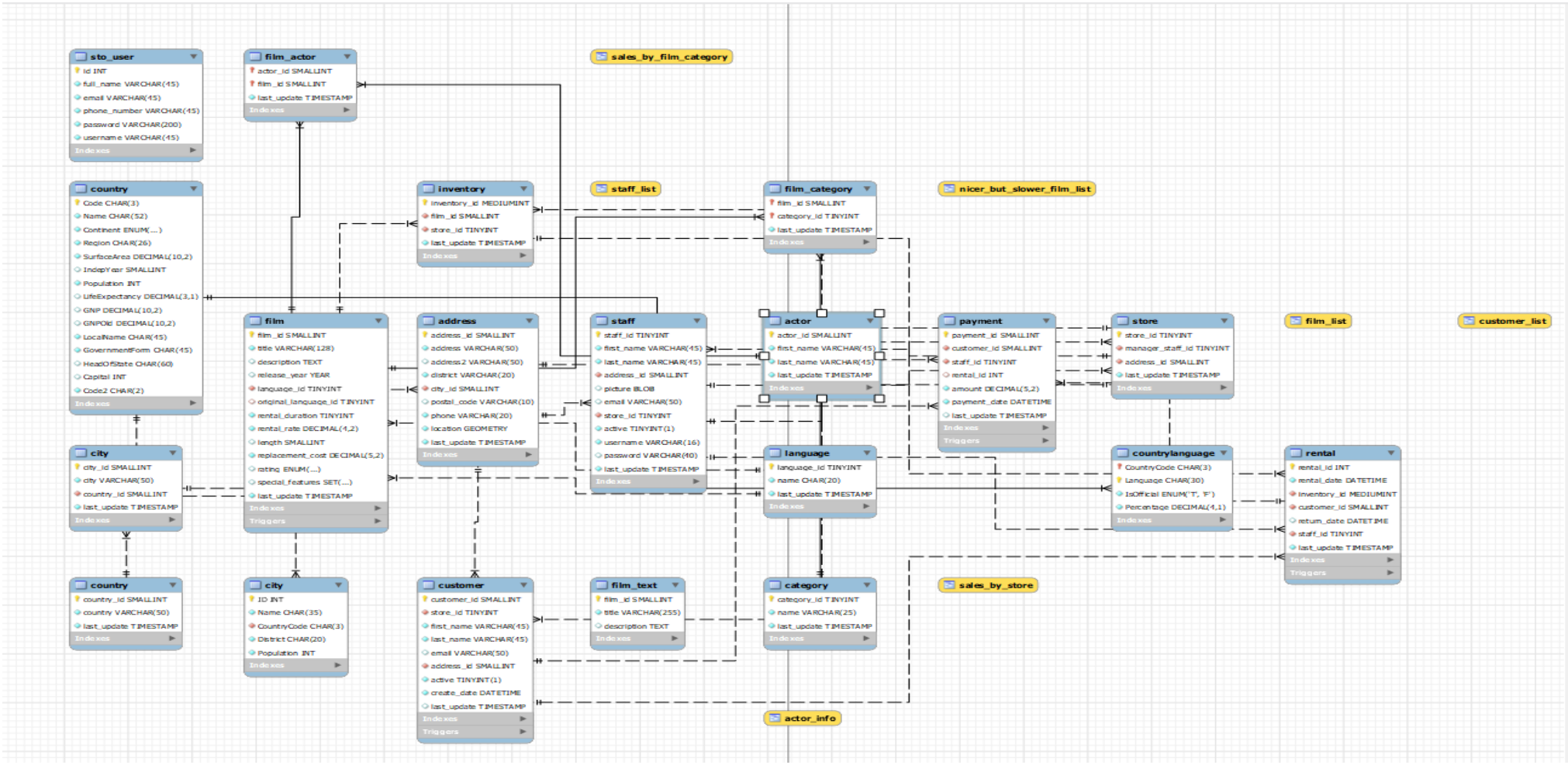

Рисунок 3.7 – ERD діаграма

Основною таблицею бази даних є "sto_user", в якій зберігаються дані користувачів. Ця таблиця містить наступні поля:

- "id" - унікальний ідентифікатор користувача.
- "full_name" - повне ім'я користувача.
- "email" - електронна пошта користувача.
- "phone_number" - номер телефону користувача.
- "password" - пароль користувача.
- "username" - ім'я користувача.

У таблиці 'sto_user' поле 'id' є первинним ключем і гарантує унікальність записів. Крім того, поля 'email', 'phone_number' та 'username' мають унікальні індекси, що гарантують унікальність значень цих полів.

ERD-діаграми сприяють розумінню інформаційних систем, забезпечуючи візуалізацію структур баз даних і зв'язків між таблицями. Такий процес проектування бази даних забезпечує ефективну організацію зберігання та доступу до даних на веб-порталі та надає необхідну функціональність і цілісність даних користувачам веб-сайту СТО "Автограф".

Field Types

#	Field	Schema	Table	Type	Character Set	Display Size	Precision	Scale
1	id	sto	sto_user	INT	binary	11	2	0
2	full_name	sto	sto_user	VARCHAR	utf8mb4	45	12	0
3	email	sto	sto_user	VARCHAR	utf8mb4	45	21	0
4	phone_number	sto	sto_user	VARCHAR	utf8mb4	45	13	0
5	password	sto	sto_user	VARCHAR	utf8mb4	200	60	0
6	username	sto	sto_user	VARCHAR	utf8mb4	45	14	0

Рисунок 3.7 – Структура БД

3.1 Налаштування безпеки

Перед початком налаштування безпеки веб-додатку були виконані наступні пункти:

- Додані необхідні залежності та бібліотеки для безпеки, зокрема Spring Security.
- Визначені ролі користувачів і їхні дозволи доступу до ресурсів додатку.

Конфігурація **Spring Security**

У класі **SecurityConfig**, який налаштовує безпеку додатку, були виконані наступні дії:

- Використано анотацію **@EnableWebSecurity** для включення безпекових функцій у веб-додатку.
- Створено клас, що розширює **WebSecurityConfigurerAdapter**, для налаштування правил безпеки.
- Визначено метод **configure**(HttpSecurity http), який встановлює правила доступу до ресурсів додатку.
- Визначено метод **configure**(AuthenticationManagerBuilder auth), який налаштовує механізм аутентифікації користувачів.

Аутентифікація та авторизація

- Використано **UserDetailsService** для зчитування даних користувачів з бази даних або іншого джерела.
- За допомогою **PasswordEncoder** забезпечено шифрування паролів користувачів для зберігання в безпечному вигляді.
- Визначено налаштування доступу до ресурсів за ролями користувачів за допомогою методу **http.authorizeRequests**().
- Використано форму аутентифікації зі стандартними поліми для введення ім'я користувача та пароля.

Захист від CSRF-атак

- Використано механізм CSRF-токенів для захисту від атак, що базуються на підробці форм.
- Встановлено параметри безпеки, пов'язані з CSRF-атаками, за допомогою методу **http.csrf().**

```
@Configuration
public class SecurityConfiguration {

    2 usages
    private CustomUserDetailsService userDetailsService;

    no usages
    public SecurityConfiguration(CustomUserDetailsService userDetailsService) {
        this.userDetailsService = userDetailsService;
    }

    no usages
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
                .csrf().disable()
                    .authorizeHttpRequests()
                    .anyRequest()
                    .permitAll()
                .and()
                    .formLogin()
                    .loginPage("/login")
                    .defaultSuccessUrl("/", true)
                    .permitAll()
                .and()
                    .logout()
                    .logoutUrl("/logout")
                    .logoutSuccessUrl("/login")
                    .invalidateHttpSession(true)
                    .deleteCookies("JESSIONID")
                    .permitAll();

        return http.build();
    }
```

Рисунок 3.1 – Конфігурація безпеки

3.2 Контролер домашньої сторінки

HomeController.java містить контролер, який відповідає за обробку HTTP-запитів і надсилання повідомлень, пов'язаних із головною сторінкою веб-ресурсу.

```
package com.example.harutadiploma.controller;

import ...

no usages
@Controller
public class HomeController {

    1 usage
    @Autowired
    private UserRepository userRepository;

    1 usage
    @Autowired
    private EmailSenderService emailSenderService;

    no usages
    @GetMapping("/")
    public String homePage(Model model, Authentication authentication) {
        model.addAttribute("userFormDto", new UserFormDto());

        if (authentication != null) {
            Optional<User> user = userRepository.findUserByUsername(authentication.getName());

            UserFormDto userFormDto = new UserFormDto();
            userFormDto.setEmail(user.get().getEmail());
            userFormDto.setPhone(user.get().getPhone());
            userFormDto.setFullName(user.get().getFullName());

            model.addAttribute("userFormDto", userFormDto);
        }
        return "index";
    }
```

Рисунок 3.2 – Контролер домашньої сторінки

- **@Controller** - анотація, що позначає клас як контролер у Spring.
- **@Autowired** - анотація, що використовуєся для автоматичного впровадження залежностей (dependency injection).

- @**GetMapping**("/") - анотація, що вказує на обробку HTTP GET-запиту на кореневому шляху.
- @**PostMapping**("/**sendMessage**") - анотація, що вказує на обробку HTTP POST-запиту на шляху "/**sendMessage**".
- Метод **homePage** контролера обробляє GET-запити для кореневого шляху. Він отримує об'єкт Authentication, що містить інформацію про авторизованого користувача. Після аутентифікації користувача дані про нього заносяться в UserFormDto і передаються шаблону для відображення на головній сторінці.

Метод **sendForm** відповідає за обробку POST-запитів для шляху "/sendMessage". Він перевіряє форму на наявність помилок (bindingResult.hasErrors()). Якщо помилки є, відбувається повернення на головну сторінку; якщо помилок немає, генерується текстове повідомлення з даними, введеними у форму, і відправляється на електронну адресу "blabla.a228@gmail.com" за допомогою **emailSenderService**.

```
no usages
@PostMapping("/sendMessage")
public String sendForm(@Valid UserFormDto userFormDto, BindingResult bindingResult, Model model) {
    if (bindingResult.hasErrors()) {
        return "index";
    }
    String messageBody = userFormDto.getFullName() + " need help with " + userFormDto.getService() + "\n" +
            "Phone number: " + userFormDto.getPhone() + "\n" + "Mail: " + userFormDto.getEmail();
    emailSenderService.sendSimpleEmail("blabla.a228@gmail.com", "New order", messageBody);
    return "redirect:/";
}
```

Рисунок 3.3 – Відправлення повідомлення на поштову адресу

3.3 Контроллер реєстрації та авторизації

Цей код відповідає за обробку запитів, пов'язаних з автентифікацією користувача: метод **login**() повертає сторінку з формою входу для введення облікових даних; метод **logout**() перенаправляє користувача на сторінку входу, після чого він виходить з системи.

Цей контролер допомагає керувати сторінками авторизації і виходу з системи в рамках мого веб-додатку.

```
package com.example.harutadiploma.controller;

import ...

no usages
@Controller
public class LoginController {

    no usages
    @GetMapping(value = {"/login"})
    public String login() { return "login"; }

    no usages
    @GetMapping({"/logout"})
    public String logout() { return "redirect:/login"; }
}
```

Рисунок 3.4 – Контролер авторизації

- @Controller - анотація, що позначає клас LoginController як контролер, який обробляє HTTP-запити.
- @GetMapping(value = {"/login"}) - анотація, що позначає метод login() як обробник GET-запитів для шляху "/login". При отриманні GET-запиту на цей шлях, метод повертає назву шаблону "login", який буде відображений на стороні клієнта.
- @GetMapping({"/logout"}) - анотація, що позначає метод logout() як обробник GET-запитів для шляху "/logout". При отриманні GET-запиту на цей шлях, метод повертає редирект на шлях "/login", що означає, що після виходу з системи користувач буде перенаправлений на сторінку входу.

```java
@Controller
public class RegistrationController {

    5 usages
    private final UserRepository userRepository;

    2 usages
    private final PasswordEncoder passwordEncoder;

    no usages
    public RegistrationController(UserRepository userRepository, PasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    no usages
    @GetMapping("/registration")
    public String saveUser(User user, Model model) { return "registration"; }

    no usages
    @PostMapping("/registration")
    public String saveUser(@Valid User user, BindingResult bindingResult, Model model) {
        Optional<User> byEmail = userRepository.findUserByEmail(user.getEmail());
        Optional<User> byPhone = userRepository.findUserByPhone(user.getPhone());
        Optional<User> byUsername = userRepository.findUserByUsername(user.getUsername());
        if(byPhone.isPresent()){
            model.addAttribute("phone_error", "This phone already used");
            return "registration";
        }
        if(byEmail.isPresent()){
            model.addAttribute("email_error", "This email already used");
            return "registration";
        }
        if(byUsername.isPresent()){
            model.addAttribute("username_error", "This username already used");
            return "registration";
        }
        if (bindingResult.hasErrors()) {
```

Рисунок 3.5 – Контролер реєстрації

Цей код відповідає за обробку запитів, пов'язаних з реєстрацією користувача. Метод saveUser(User user, Model model) виконується при GET-запиті на шлях "/registration" і повертає сторінку з формою реєстрації. Метод saveUser(@Valid User user, BindingResult bindingResult, Model model) виконується при POST-запиті на шлях "/registration" і здійснює перевірку та збереження нового користувача, якщо всі дані валідні.

@Controller - анотація, що позначає клас RegistrationController як контролер, який обробляє HTTP-запити.

@GetMapping("/registration") - анотація, що позначає метод saveUser(User user, Model model) як обробник GET-запитів для шляху "/registration". При отриманні GET-запиту на цей шлях, метод повертає назву шаблону "registration", який буде відображений на стороні клієнта.

@PostMapping("/registration") - анотація, що позначає метод saveUser(@Valid User user, BindingResult bindingResult, Model model) як обробник POST-запитів для шляху "/registration". При отриманні POST-запиту на цей шлях, метод перевіряє користувачів на наявність за допомогою userRepository і валідує отримані дані. Якщо дані недійсні або користувач уже існує, помилки додаються до моделі model, а метод повертає назву шаблону "registration". В іншому випадку, створюється новий користувач, його пароль шифрується, і він зберігається у userRepository. Після цього метод перенаправляє користувача на шлях "/login".

3.4 Клас моделі User для представлення користувача

Клас User визначає структуру користувача з такими полями, як id, username, fullName, password, email та phone. Кожне поле має свої обмеження, що вказані за допомогою анотацій валідації. Наприклад, username повинен бути не порожнім рядком довжиною від 4 до 16 символів, password повинен бути не порожнім рядком довжиною не менше 6 символів, а email повинен відповідати формату електронної пошти. Клас також містить анотацію @Table, що вказує назву таблиці бази даних, в яку будуть зберігатися об'єкти User.

```java
public class User {

    no usages
    @Id
    private int id;

    no usages
    @Column
    @NotBlank(message = "Username is mandatory")
    @Size(min = 4, max = 16, message = "Size must be in 4-16")
    private String username;

    no usages
    @Column
    @NotBlank(message = "Full name is mandatory")
    private String fullName;

    no usages
    @Column
    @NotBlank(message = "Password is mandatory")
    @Size(min = 6, message = "Size must be min 6")
    private String password;

    no usages
    @Column
    @Email
    private String email;

    no usages
    @Column(name = "phone_number")
    @NotBlank(message = "Phone is mandatory")
    @Size(min = 6, max = 15, message = "Size must be in 6-20")
    private String phone;
}
```

Рисунок 3.6 – Код моделі User

- @Entity - анотація, що позначає клас User як сутність бази даних, яка буде зберігатися у таблиці.
- @Table(name = "sto_user") - анотація, що вказує назву таблиці бази даних, в яку будуть зберігатися об'єкти User.

- @Data - анотація з бібліотеки Lombok, яка автоматично генерує методи геттерів, сеттерів, toString, equals і hashCode для класу User.
- @Id - анотація, що позначає поле id як первинний ключ таблиці.
- @Column - анотація, що вказує, що поле відповідає стовпцю таблиці бази даних.
- @NotBlank - анотація валідації, яка вимагає, щоб поле не було порожнім або не містило тільки пробіли.
- @Size - анотація валідації, яка вказує діапазон розміру для поля.
- @Email - анотація валідації, яка перевіряє, чи поле містить правильний формат електронної пошти.

ВИСНОВКИ

Переддипломна робота дала значні результати в розробці інформаційної системи для сайту СТО "Автограф". Проект був зосереджений на створенні функціональної та ефективної платформи для автосервісу, яка дозволяє зручно та ефективно управляти клієнтами, замовленнями та іншими важливими аспектами діяльності автосервісу.

Система була розроблена з використанням потужних інструментів і технологій, включаючи одну з найпопулярніших мов програмування Java та фреймворків, таких як Spring Framework. Крім того, в якості системи управління базами даних було обрано MySQL, яка дозволяє ефективно та безпечно зберігати та обробляти великі обсяги інформації.

Під час розробки особливу увагу було приділено архітектурним принципам та належним практикам програмування, що сприяло створенню стабільної, масштабованої та простої у використанні системи. Система виконувала функції, необхідні для ефективної роботи автосервісу, такі як аутентифікація та авторизація користувачів, управління замовленнями, а також зберігання та відображення інформації про клієнтів.

В результаті цього проекту створено інформаційну систему, яка оптимізує роботу, покращує обслуговування клієнтів і впорядковує бізнес-процеси. Реалізація цього проекту дала цінний практичний досвід у розробці програмного забезпечення та використанні сучасних технологій.

СПИСОК ЛІТЕРАТУРИ

1. Admin. Переваги PHP - 10 найважливіших переваг PHP. *Education-WIKI.com*. URL: https://uk.education-wiki.com/3404140-advantages-of-php (дата звернення: 06.06.2023).
2. Python: преимущества и недостатки языка Python - блог IT-школы Hillel. *Корисні матеріали: Статті та новини IT-індустрії | Комп'ютерна школа Hillel*. URL: https://blog.ithillel.ua/ru/articles/preimushchestva-i-nedostatki-yazyka-python (дата звернення: 06.06.2023).
3. Мова програмування Ruby: чи варто вивчати, плюси і мінуси | #ТЕГ. *#ТЕГ |*. URL: http://teg.com.ua/mova-programyvannia-ruby-chi-varto-vivchati-plusi-i-minysi/ (дата звернення: 06.06.2023).
4. Переваги та недоліки Java - From-UA: Новости Украины. *From-UA: Новости Украины*. URL: https://from-ua.info/perevahy-ta-nedoliky-java/ (дата звернення: 06.06.2023).
5. Фреймворк Spring та його особливості. *Highload.today - медиа для разработчиков*. URL: https://highload.today/uk/frejmvork-spring-ta-jogo-osoblivosti/ (дата звернення: 11.06.2023).
6. Bender. Eclipse, NetBeans Ñ‡ Ð¸ IntelliJ IDEA? Ðž Ð±Ð¸Ñ€Ð°Ñ” Ð¼Ð¾ IDE Ð´Ð»Ñ • Java-Ñ€ Ð¾Ð·Ñ€Ð¾Ð±Ð½Ð¸ÐºÐ°. *JavaRush*. URL: https://javarush.com/ua/groups/posts/4027-eclipse-netbeans-ili-intellij-idea-vihbiraem-ide-dlja-java-razrabotki (дата звернення: 06.06.2023).
7. СУБД (Системи управління базами даних) - що це, види та функції. *Highload.today - медиа для разработчиков*. URL: https://highload.today/uk/subd-yaki-buvayut-yak-vibrati/ (дата звернення: 06.06.2023).

8. Walls C. Spring in action. 2nd ed. Greenwich, CT : Manning Publications Co., 2008. 730 p.
9. MySQL - офіційна документація. *mysql.com*. URL: https://dev.mysql.com/doc/ (дата звернення: 06.06.2023).
10. Java Documentation - Get Started. *Oracle Help Center*. URL: https://docs.oracle.com/en/java/ (дата звернення: 06.06.2023).
11. Bloch J. Effective Java. Pearson Education, Limited, 2017.
12. Bauer C., King G. Java Persistence mit Hibernate. München : Carl Hanser Verlag GmbH & Co. KG, 2007. URL: https://doi.org/10.3139/9783446413825 (дата звернення: 06.06.2023).
13. Scarioni C., Nardone M. Pro Spring Security. Berkeley, CA : Apress, 2019. URL: https://doi.org/10.1007/978-1-4842-5052-5 (дата звернення: 08.06.2023).
14. Books G. Spring Into Action!. Golden Books, 2007. 48 c.
15. Dubois P. Mysql Cookbook. O'Reilly Media, Incorporated, 2006.
16. Naughton P., Schildt H. Java: The Complete Reference (Complete Reference Series). Mcgraw-Hill Osborne Media, 1996. 886 p.
17. Bloch J. Effective Java. Pearson Education, Limited, 2017.
18. Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall, 2008.
19. Documentation - 6.2 - Hibernate ORM. *Hibernate*. URL: https://hibernate.org/orm/documentation/6.2/ (дата звернення: 08.06.2023).

ДОДАТОК

1.Лістинг програми

```
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.example.harutadiploma.config;

import com.example.harutadiploma.user.CustomUserDetailsService;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configurers.AuthorizeHttpRe
questsConfigurer;
import
org.springframework.security.config.annotation.web.configurers.FormLoginConfi
gurer;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

@Configuration
public class SecurityConfiguration {
   private CustomUserDetailsService userDetailsService;

   public                    SecurityConfiguration(CustomUserDetailsService
userDetailsService) {
```

```
        this.userDetailsService = userDetailsService;
    }

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception
{

((HttpSecurity)((FormLoginConfigurer)((FormLoginConfigurer)((HttpSecurity)((
AuthorizeHttpRequestsConfigurer.AuthorizedUrl)((HttpSecurity)http.csrf().disable
()).authorizeHttpRequests().anyRequest()).permitAll().and()).formLogin().loginPa
ge("/login").defaultSuccessUrl("/",
true)).permitAll()).and()).logout().logoutUrl("/logout").logoutSuccessUrl("/login"
).invalidateHttpSession(true).deleteCookies(new
String[]{"JESSIONID"}).permitAll();
        return (SecurityFilterChain)http.build();
    }

    @Bean
    protected PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    protected DaoAuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider daoAuthenticationProvider = new
DaoAuthenticationProvider();

daoAuthenticationProvider.setPasswordEncoder(this.passwordEncoder());

daoAuthenticationProvider.setUserDetailsService(this.userDetailsService);
        return daoAuthenticationProvider;
    }
}
//
// Source code recreated from a .class file by IntelliJ IDEA
```

```
// (powered by FernFlower decompiler)
//

package com.example.harutadiploma.controller;

import com.example.harutadiploma.model.User;
import com.example.harutadiploma.model.UserFormDto;
import com.example.harutadiploma.repository.UserRepository;
import com.example.harutadiploma.service.EmailSenderService;
import jakarta.validation.Valid;
import java.util.Optional;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class HomeController {
   @Autowired
   private UserRepository userRepository;
   @Autowired
   private EmailSenderService emailSenderService;

   public HomeController() {
   }

   @GetMapping({"/"})
   public String homePage(Model model, Authentication authentication) {
      model.addAttribute("userFormDto", new UserFormDto());
```

```
            if (authentication != null) {
                Optional<User> user = this.userRepository.findUserByUsername(authentication.getName());
                UserFormDto userFormDto = new UserFormDto();
                userFormDto.setEmail(((User)user.get()).getEmail());
                userFormDto.setPhone(((User)user.get()).getPhone());
                userFormDto.setFullName(((User)user.get()).getFullName());
                model.addAttribute("userFormDto", userFormDto);
            }

            return "index";
        }

        @PostMapping({"/sendMessage"})
        public String sendForm(@Valid UserFormDto userFormDto, BindingResult bindingResult, Model model) {
            if (bindingResult.hasErrors()) {
                return "index";
            } else {
                String var10000 = userFormDto.getFullName();
                String messageBody = var10000 + " need help with " + userFormDto.getService() + "\nPhone number: " + userFormDto.getPhone() + "\nMail: " + userFormDto.getEmail();

this.emailSenderService.sendSimpleEmail("harutasender@gmail.com", "New order", messageBody);
                return "redirect:/";
            }
        }
    }
    //
    // Source code recreated from a .class file by IntelliJ IDEA
    // (powered by FernFlower decompiler)
```

```
//

package com.example.harutadiploma.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
public class LoginController {
    public LoginController() {
    }

    @GetMapping({"/login"})
    public String login() {
        return "login";
    }

    @GetMapping({"/logout"})
    public String logout() {
        return "redirect:/login";
    }
}
//
// Source code recreated from a .class file by IntelliJ IDEA
// (powered by FernFlower decompiler)
//

package com.example.harutadiploma.controller;

import com.example.harutadiploma.model.User;
import com.example.harutadiploma.repository.UserRepository;
import jakarta.validation.Valid;
```

```
import java.util.Optional;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;

@Controller
public class RegistrationController {
    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;

    public RegistrationController(UserRepository userRepository, PasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    @GetMapping({"/registration"})
    public String saveUser(User user, Model model) {
        return "registration";
    }

    @PostMapping({"/registration"})
    public String saveUser(@Valid User user, BindingResult bindingResult, Model model) {
        Optional<User> byEmail = this.userRepository.findUserByEmail(user.getEmail());
        Optional<User> byPhone = this.userRepository.findUserByPhone(user.getPhone());
        Optional<User> byUsername = this.userRepository.findUserByUsername(user.getUsername());
```

```
            if (byPhone.isPresent()) {
                model.addAttribute("phone_error", "This phone already used");
                return "registration";
            } else if (byEmail.isPresent()) {
                model.addAttribute("email_error", "This email already used");
                return "registration";
            } else if (byUsername.isPresent()) {
                model.addAttribute("username_error", "This username already used");
                return "registration";
            } else if (bindingResult.hasErrors()) {
                return "registration";
            } else {
                User newUser = new User();

newUser.setPassword(this.passwordEncoder.encode(user.getPassword()));
                newUser.setPhone(user.getPhone());
                newUser.setEmail(user.getEmail());
                newUser.setFullName(user.getFullName());
                newUser.setUsername(user.getUsername());
                this.userRepository.save(newUser);
                return "redirect:/login";
            }
        }
    }
    //
    // Source code recreated from a .class file by IntelliJ IDEA
    // (powered by FernFlower decompiler)
    //

    package com.example.harutadiploma.model;
```

```
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import jakarta.validation.constraints.Email;
import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Size;

@Entity
@Table(
    name = "sto_user"
)
public class User {
    @Id
    private int id;
    @Column
    private @NotBlank(
    message = "Username is mandatory"
) @Size(
    min = 4,
    max = 16,
    message = "Size must be in 4-16"
) String username;
    @Column
    private @NotBlank(
    message = "Full name is mandatory"
) String fullName;
    @Column
    private @NotBlank(
    message = "Password is mandatory"
) @Size(
    min = 6,
```

```
    message = "Size must be min 6"
) String password;
    @Column
    private @Email String email;
    @Column(
        name = "phone_number"
    )
    private @NotBlank(
    message = "Phone is mandatory"
) @Size(
    min = 6,
    max = 15,
    message = "Size must be in 6-20"
) String phone;

    public User() {
    }

    public int getId() {
        return this.id;
    }

    public String getUsername() {
        return this.username;
    }

    public String getFullName() {
        return this.fullName;
    }

    public String getPassword() {
        return this.password;
```

```
    }

    public String getEmail() {
        return this.email;
    }

    public String getPhone() {
        return this.phone;
    }

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

    public void setUsername(final String username) {
        this.username = username;
    }

    public void setFullName(final String fullName) {
        this.fullName = fullName;
    }

    public void setPassword(final String password) {
        this.password = password;
    }

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

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

```
}

public boolean equals(final Object o) {
    if (o == this) {
        return true;
    } else if (!(o instanceof User)) {
        return false;
    } else {
        User other = (User)o;
        if (!other.canEqual(this)) {
            return false;
        } else if (this.getId() != other.getId()) {
            return false;
        } else {
            label73: {
                Object this$username = this.getUsername();
                Object other$username = other.getUsername();
                if (this$username == null) {
                    if (other$username == null) {
                        break label73;
                    }
                } else if (this$username.equals(other$username)) {
                    break label73;
                }

                return false;
            }

            Object this$fullName = this.getFullName();
            Object other$fullName = other.getFullName();
            if (this$fullName == null) {
                if (other$fullName != null) {
```

```
            return false;
        }
    } else if (!this$fullName.equals(other$fullName)) {
        return false;
    }

    label59: {
        Object this$password = this.getPassword();
        Object other$password = other.getPassword();
        if (this$password == null) {
            if (other$password == null) {
                break label59;
            }
        } else if (this$password.equals(other$password)) {
            break label59;
        }

        return false;
    }

    Object this$email = this.getEmail();
    Object other$email = other.getEmail();
    if (this$email == null) {
        if (other$email != null) {
            return false;
        }
    } else if (!this$email.equals(other$email)) {
        return false;
    }

    Object this$phone = this.getPhone();
    Object other$phone = other.getPhone();
```

```
            if (this$phone == null) {
                if (other$phone != null) {
                    return false;
                }
            } else if (!this$phone.equals(other$phone)) {
                return false;
            }

            return true;
        }
    }
}

protected boolean canEqual(final Object other) {
    return other instanceof User;
}

public int hashCode() {
    int PRIME = true;
    int result = 1;
    result = result * 59 + this.getId();
    Object $username = this.getUsername();
    result = result * 59 + ($username == null ? 43 : $username.hashCode());
    Object $fullName = this.getFullName();
    result = result * 59 + ($fullName == null ? 43 : $fullName.hashCode());
    Object $password = this.getPassword();
    result = result * 59 + ($password == null ? 43 : $password.hashCode());
    Object $email = this.getEmail();
    result = result * 59 + ($email == null ? 43 : $email.hashCode());
    Object $phone = this.getPhone();
    result = result * 59 + ($phone == null ? 43 : $phone.hashCode());
    return result;
```

```
    }

    public String toString() {
        int var10000 = this.getId();
        return "User(id=" + var10000 + ", username=" + this.getUsername() + ", fullName=" + this.getFullName() + ", password=" + this.getPassword() + ", email=" + this.getEmail() + ", phone=" + this.getPhone() + ")";
    }
}
```