МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Автоматизована інформаційна система онлайн-консультацій для університетської клініки СумДУ»

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

Керівник роботи — **Берест О.Б.**

Студента групи ІН – 62 — **Полушкіної І.О.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 г.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-62 спеціальності "Інформаційно-комунікаційні технології" денної форми навчання Полушкіної Інни Олександрівни.

Тема: "Автоматизована інформаційна система онлайн-консультацій для університетської клініки СумДУ"

Затверджена наказом по СумДУ

№_____ от _____2020 г.

Зміст пояснювальної записки: 1) огляд аналогічних систем; 2) огляд типу архітектури системи, патернів проектування, вибір типу програмного забезпечення; 3) вибір технологій для розробки проекту, серед яких: мова програмування, база даних, фреймвокри;5) результат розробки проекту; 6) шдяхи подальшого розвитку інформаційної системи.

Дата видачі завдання "_____"_____2020 г.

Керівник випускної роботи _____ Берест О.Б.

Завдання приняв до виконання _____ Полушкіна І.О.

РЕФЕРАТ

Записка: 57 стор., 14 рис., 1 табл., 20 джерел.

Об’єкт дослідження — система для онлайн запису на консультацію.

Мета роботи — розробка інформаційної системи для онлайн запису на консультацію до лікаря.

Результати — розроблено веб-додаток автоматизованої інформаційної системи для онлайн запису на консультацію до лікаря, що дає можливість користувачам заплановувати консультації на основі даних про працівників та послуги клініки, працівники клініки мають можливість редагувати свій графік роботи, користувачі та працівники мають можливість переглядати усі свої консультації.

Ключові слова: інформаційна система, система, веб-додаток, проект, фреймворк, база даних.

Зміст

Вступ

У сучасному світі зі швидким темпом життя технології все більше стають популярними для користувачів, що намагаються заощадити час. Сучасні інформаційні технології спрошують життя їх користувачам, особливо якщо це стосується швидкості та якості обслуговування в медичній сфері.

Системи для планування часу є важливим інструментом для керівників, оскільки допомогають розпланувати робочий час для персоналу, централізувати дані користувачів.

Враховуючи все більш активне впровадження інформаційних технологій в повсякдення життя людей, одним із ефективних рішень для планування часу є створення автоматизованої інформаційної системи для онлайн запису на консультацію до лікаря.

Нашою метою є створення зручної автоматизованої інформаційної системи, що допоможе оптимізувати час її користувачам – клієнтам клініки та медичному персоналу.

Подібні системи мають багато переваг не тільки для клієнтів, а також і для спеціалістів та їх керівників. Дана система допоможе заощадити час для запису на консультацію, централізує дані – створить список консультацій для користувачів, допоможе розпланувати графік роботи для персоналу клініки, допоможе побачити різницю між запланованими проведеними консультаціями та непроведеними. Клієнти не стояти в черзі, а прийти на завчасно заплановану консультацію до потрібного їм лікаря. За допомогою такої системи медичний центр зможе оперувати усіма клієтськими даними та покарщувати якість обслуговування.

1 Аналіз предметної області

Медична сфера тісно пов’язана з життям з нашим життя. Кожен хоче будти здорови на вчасно отримувати необхідну медичну допомогу. Інформаційна система для онлайн запису на консультацію до лікаря допоможе вчасно та зручно отримувати медичну допомогу.

1.1 Огляд існуючих рішень

Оглянемо найбільш популярні аналогічні рішення, що можуть покращити функціонування університетської клініки СумДУ.

Найбільш відомим програмним забезпеченням, що може бути використане для схожих цілей є:

1) YCLIENTS – онлайн система підтримки бізнесу, що потребує онлайн запис на консультацію або зустріч. Може застосовуватися у багатьох сферах діяльності, таких як медицина, спорт, побутові послуги, освіта та ін.

За допомогою даної системи клієнти можуть швидко та зручно обрати потрібний для них час для запису на консультацію. Клієнти маю можливість самі налагоджувати графік, отримувати повідомленні від адміністратора перед консультацією, про зміни в ній, про нові послуги, можливості та ін. Клієнти можуть керувати своїм графіком онлайн через веб-сайт або через мобільний додаток. На основі даної системи можна легко створити мобільні додатки як для Android OS, так і для iOS.

YCLIENTS дозволяє контролювати зміни внесені адміністратором, що спрощує пошук та виправлення помилок. Система дозволяє керувати фінансовими потоками, показує відразу клієнту вартість обраної ним послуги чи консультації.

Система дозволяє бачити статистику бізнес послуг, розраховувати заробітну плату персоналу, вести облік витрат та оптимізувати їх використання.

YCLIENTS має безкоштовний пробний період – 7 днів, після чого користування стає платним, біля 900 грн за місяць. Цінова політика системи залежить від кількості працівників компанії.

2) Appointer – система для онлайн запису на зустрічі. Дозволяє легко створювати та редагувати записи на зустрічі, керувати графіком працівників та бачити статистику навантаження. Клієнти мають можливість самостійно записуватися на зустрічі у зручних для них час, незадовго до зустрічі система повідомить клієнта за допомогою СМС.

Appointer дає можливість керувати клієнтською базою, бачити статистику зустрічей, бачити відсоток нових клієнтів та відсоток втрачених клієнтів. Дозволяє автоматично розраховувати заробітну плату робітникам. Система дає можливість бачити використані матеріли та надлишкові.

Детальна статистика, яка дає повну картину про роботу підприємства:

- Прибуток компанії;
- Кількість записів на прийом по будь-яким періодам, з можливістю порівняння аналогічних періодів по місяцях і роках;
- Завантаженість фахівців і залів, з можливістю порівняння за періодами;
- Рентабельність залів і працівників;
- Популярні послуги і товари.

Має безкоштовну двотижневу версію, після чого користування стає платним, біля 500 грн за місяць в залежності від обраного тарифу користування. Цінова політика залежить від часу користування системою.

3) SimplyBook.me – система онлайн бронювання для сфери послуг, навчальних, медичних та інших закладів, що потребують попереднього запису. Дана система має у собі оптимізований для мобільних пристроїв веб-сайт, що дозволяє керувати записом клієнтів на зустрічі та редагувати їх за необхідності.

Клієнти можуть записувати на консультації у зручний для них, редагувати свій графік, отримувати нагадування про заплановану зустріч. Також є можливість запису одразу групи людей або одразу запланувати кілька зустрічей – це є особливо актуальним для навчальних та медичних закладів.

Після проведеної консультації клієнти мають можливість залишити відгук.

Однією з переваг даної системи є безпека при передачі даних, дані є зашифрованими під час передачі. Також задля безпеки та збереження даних відбувається щоденне резервне копіювання.

SimplyBook спрощує спілкування з клієнтами, за допомогою нагадування їм про заплановані зустрічі, системи відгуків та системи повідомлення для персоналу.

Система надає безкоштовний тестовий період, після якого користування стає платним, біля 800 грн за місяць, в залежності від тарифу.

Вищеописані системи мають великий ряд переваг, серед яких можливості для зручної комунікації з клієнтами, ведення грошової статистики витрат та їх оптимізацію та багато іншого. Але й мають свої недоліки, що робить неможливим їх використання для університетської клініки СумДУ. Основний недолік – цінова політика запропонованих систем. Для системи YCLIENTS є недоцільним використання мобільних веб додатків для клієнтів, краще використовувати оптимізовану під мобільні пристрої версію веб-сайту.

Отже, пропонуємо створити власну автоматизовану систему для онлайн запису на консультації, що буде відповідати таким вимогам:

Таким чином постає задача проаналізувати сучасні системи онлайн запису та розробити аналогічне програмне забезпечення, яке буде :

1) Web-додатком;
2) Дозволятиме клієнтам записуватись на консультацію онлайн, редагувати її за бажанням;
3) Дозволятиме працівникам клініка редагувати графік проведення консультацій;
4) Використовувати базу даних для зберігання інформації про консультації та їх учасників;
5) Мати простий та зрозумілий інтерфейс користувача, який не потребує додаткових технічних знань .

2 Огляд технологій для вирішення проблеми

Автоматизована інформаційна система онлайн консультацій має відповідати на запити від багатьох клієнтів одночасно, надаючи їм необхідну інформацію або фіксуючи внесені ними зміни у розклад консультацій. Тобто багато клієнтів повинні мати одночасний доступ до ресурсів системи. Інформаційна система має чекати на запити від клієнтів, оброблювати їх та давати клієнту відповідь на його запит.

Також інформаційна система має оперувати великою кількістю даних про минулі консультації, різні варіанти консультацій, що можуть бути заплановані користувачами у майбутньому, зберігати дані про заплановані консультації, та оновлювати їх. На основі оброблених даних система має створювати та надсилати своїм користувачам повідомлення про заплановану консультацію або внесення змін до неї.

2.1 Вибір архітектурної моделі системи

Для того щоб розроблювана інформаційна система могла добре функціонувати довгий час, вона має забезпечувати поставленим вимогам, мати високу якість та бути доступною на легкою у використанні для звичайних користувачів. На всі ці параметри впливає архітектура програмного забезпечення.

Архітектура – це базова організація системи, втілена в її компонентах, зовнішніми та внутрішніми відношеннями у системі, а також принципи, що визначають проектування та розвиток системи.[1]

Правильно та своєчасно спроектована архітектура полегшить подальшу розробку, впровадження, користування, модифікацію інформаційної системи.

Для створення подібних систем доцільно використовувати клієнт-серверну архітектуру програмного забезпечення.

Сучасний підхід клієнт-сервер передбачає сервер, що здебільшого забезпечує доступ до ресурсів (як правило комунікація між клієнтом і сервером передаються у XML або JSON) у відповідь на запити клієнта. Для

сучасної клієнт-серверної моделі часто використовуються асинхронні запити до сервера. Сервер спочатку обслуговує сторінки з малою кількістю даних. Сторінки, реагуючи на дії користувача, роблять асинхронні запити до сервера і сервер просто реагує на ці повідомлення, що викликає поточне оновлення сторінки.[2]

З часом розвиваються технології та змінюються потреби користувачів, це змушую розробників змінювати архітектурні моделі. Сьогодні веб-розробник повинен використовувати інструменти та підхід до розробки, що узгоджуються із сучасною веб-сценою. Наразі все більше розробників використовують клієнт-серверний підхід тому, що він дозволяє робити архітектуру додатку більш гнучкою. Завдяки цьому при необхідності змін – редагування або додавання, можуть бути зроблені набагато швидше.[3]

Модифікація – це критерій легкості, з якою можна вносити зміни в архітектуру додатка. Навіть якщо можна було б створити програмну систему, що буде ідеально відповідати вимогам її користувачів, ці вимоги можуть змінюватися з часом подібно до того, як з часом змінюється суспільство. Оскільки компоненти додатку розподілені на декілька організаційних частин, система буде готова до часткових та поступових змін, коли існують декілька реалізацій, які не заважають одна одній, але дозволяють використовувати розширені можливості.[2]

З розвитком нашої системи буде набагато простіше вносити зміни у додаток, що створений з використання клієнт-серверної архітектури. Це дасть можливість клієнтській стороні обмінюватися повідомлення з сервером не вникаючи у подробиці його реалізації, а також зробить можливим внесення змін тільки для однієї сторони, не змінюючи сам механізм обміну повідомленнями.

Явною перевагою використання даного підходу є можливість розділити код на логічно подібні частини – це дає вищу згуртованість як в оригінальній конструкції, так і в постійній підтримці будь-якої системи. Чіткий поділ між клієнтською та серверною частинами робить код модульним та швидко керованим. Крім того, дані та розмітка відображення можуть бути більш чітко

відокремлені. Гнучкість та повторне використання коду - логічний результат хорошої організації коду. Повторне використання коду вірогідніше, коли є чіткі компоненти. Як мінімум, ті самі REST API може використовуватися для подання даних у різноманітні браузери та мобільні пристрої.[3]

Архітектурна модель клієнт-сервер добре підходить для створення автоматизованої інформаційної системи для онлайн консультацій. Дана модель зможе забезпечити швидку комунікацію між користувачем та працівниками клініки, завдяки розподілу на клієнтську та серверну частини. За допомогою такого підходу код системи буде розділений на модулі, наслідком чого буде його структурованість, легко керованість та можливість швидкого внесення змін.

2.2 Огляд патернів проектування

Патерн проектування – це загальне рішення ревної проблеми в дизайні архітектури. Патерн являє собою не конкретний код, а загальну концепцію або приклад рішення яке має бути налаштоване під потреби вашої програми.[4]

Використання перевірених патернів при проектуванні архітектури інформаційної системи полегшить подальший розвиток системи. Завдяки можливості повторного використання коду, яку дають нам патерни проектування, майбутнє розширення системи або внесення змін до неї будуть відбуватися швидко та з мінімальними витратами.

Оглянемо патерни проектування, що можуть бути використані для створення нашої інформаційної системи.

Клієнтська частина інформаційної системи має відповідати за обробку дій користувача. У відповідь на введену користувачем інформацію, формується та надсилається запит до серверної частини, відповідь сервера оброблюється та надається користувачу. Тобто клієнтська частина взаємодіє з клієнтом, відображаючи йому необхідну інформацію у зручному та зрозумілому для нього вигляді.

Для проектування системи можемо використати патерн проектування MVC.

Патерн Model View Controller(MVC) – шаблон проектування, основною ідеєю якого є розподілення даних додатку, користувальницького інтерфейсу та бізнес логіки на три окремих компоненти – Модель, Представлення та Контролер – таким чином модифікація кожного компоненту може здійснюватися незалежно від інших компонентів системи.[5]

Модель відображає поведінку, логіку та правила керування даними, являється незалежною від інтерфейсу користувача.

Контролер оброблює дії користувача, надаючи необхідні дані моделі або представленню.

Представлення являє собою будь-який користувальницький інтерфейс, що буде надавати контролеру необхідну інформацію.

Для розроблення клієнтської частини доцільно використати патерн проектування MVC. Це допоможе розділити відображення даних від їх обробки та зробить компоненти незалежними один від одного.

Як і для клієнтської частини, так і для серверної частини буде доцільним використання патерну MVC для розподілення бізнес логіки серверу, контролеру для обробки запитів від клієнта та повернення результату запиту. Також цей патерн допоможе правильно створити журнал дій сервера, що полегшить підтримку роботи та подальше розширення системи.

Інформаційна система має оперувати великою кількістю клієнтських даних, для збереження яких будемо використовувати базу даних. Сервер додатку має взаємодіяти з базою даних та багатьма клієнтами одночасно.

Взаємодіє сервера і бази даних повинна відбуватися за допомогою єдиного екземпляру спеціалізованого класу. Створити такий екземпляр можна використовуючи патерн Singleton.

Singleton – твірний патерн проектування, визначає клас що може мати лише один екземпляр та створює глобальну точку доступу до нього. Зазвичай використовується для доступу до ресурсу загального користування, наприклад до бази даних. [6]

Для взаємодії сервера та бази даних будемо використовувати паттерн Singleton, оскільки його використання дозволить створити єдиний екземпляр

деякого класу, що буде взаємодіяти з базою даних. Це означає, що ми матимемо єдину точку доступу до клієнтської бази даних. Патерн Singleton дозволить одночасно багатьом клієнтам взаємодіяти через сервер з базою даних не руйнуючи її єдності, кожному будуть подані актуальні дані, а внесені зміни будуть швидко застосовуватись, що зробить користування систему ефективним та зручним.

2.3 Вибір типу програмного забезпечення системи

Автоматизована інформаційна система для запису на консультації має являти собою сервіс для керування даними клієнтів. Нам необхідно створити просту на зручну у використанні систему.

Сучасна людина у повсякденному житті використовує багато різних інформаційних пристроїв, аби зробити зручнішим своє життя. Варіантами доступу до будь-яких інформаційних систем можуть бути: веб, десктоп або мобільний додаток.

Веб-додаток – це клієнт-серверний додаток, у якому клієнт взаємодіє з сервером за допомогою веб-браузера. Логіка веб-додатку розподілена між клієнтом і сервером, обмін даними між якими здійснюється за допомогою комп’ютерної мережі. Більша частина даних зберігається на сервері, або в базі даних, що взаємодіє з ним, сервер надає клієнту інформації у відповідь на запити. Таким чином на клієнтському пристрої може зберігатись лише мінімально необхідний набір даних, це являється перевагою для клієнта, адже витрати пам’яті його просторою є не значними. Це також робить веб-додаток незалежним від операційної системи, тобто не моє необхідності переробляти додаток під багато популярних операційних систем, а достатньо створити лише один раз.

Мобільний додаток – це програмне забезпечення, призначене для роботи з мобільних пристроїв. Має бути розробленим під конкретну платформу. Працює на основі клієнт-серверної архітектури, але зберігає значну частину даних на клієнтському пристрої. Використовує більше клієнтських ресурсів

ніж веб-додаток. Зазвичай мобільний додаток необхідний для системи, що має часто взаємодіяти з клієнтом.

Десктоп додаток – це програма, що працює на клієнтському комп'ютері. Може мати клієнт-серверну архітектуру, що взаємодіє за допомогою мережі, але може й працювати самостійно без доступу до мережі. Використовує багато клієнтських ресурсів, використовується, якщо є обмеження безпеки використання. При проектуванні необхідно вирішити проблему залежності від платформи.

Для створення нашої інформаційної системи краще обрати веб-додаток. Цей варіант має ряд вищеописаних переваг перед іншими, саме він буде найбільш зручним у використанні.

3 Огляд технологій реалізації проекту

Під час проектування було вирішено створити веб-додаток для інфомаційної системи онлайн запису на консультацію. Веб-додаток має бути незалежним від операційної системи, тобто при необхідності внесення змін у додаток, його не потрібно будет перероблювати під багато популярних операційних систем, а достатньо внести зміни лише один раз.

Архітектура проекту буде розділена на клієнтську та серверну частини, оскільки дана модель зможе забезпечити швидку комунікацію між користувачем та працівниками клініки. За допомогою такого підходу система буде структурована, легко керована та матиме можливість швидкого внесення змін.

Патерном, на основі якого побудована інформаційна веб-система, було обрано патерн MVC(Model-View-Controller). За допомогою даного патерну можно розділити систему на три компоненти, кожен з яких буде виконувати свою функцію. Використання даного патерну допоможе розділити відображення даних від їх обробки та зробить компоненти незалежними один від одного.

На основі раньше спроектованої архітектури потрібно обрати технології, що будуть використані під час веб-розробки.

Основні аспекти, які необхідно оглянути при виборі технологій реализіції:

- мова програмування;
- провайдера бази даних;
- фреймворки, які можуть бути використані для прискорення створення програми.

3.1 Аналіз мов програмування

Однією з найважливіших частин аналізу є вибір мови прогрмавування. Мова програмування – це штучна мова, що використовується для передачі команд комп'ютерам. Це система позначень для опису алгоритмів та структур даних, на основі яких буде працювати програмна система. Від вибору мови програмування можуть залежати основні критерії роботи системи, такі як: швидкодія, гнучкість до змін, розширення системи та ін.

Для реалізації інформаційної онлайн системи було вирішено створити веб-додаток, отже оглянемо найпопулярніші мови для веб-програмування.
Топ мов програмування для веб розробки у 2020 році [7]:

1. Python.
2. Java.
3. JavaScript.
4. PHP.
5. Go.

Розгянемо бішь докладно вищеописані мови. Давайте розглянемо лише 3 з них, які найбільш підходять для розробки веб-додатку для онлайн запису на консультації.

Python – одна з найпопулярніших мов програмування в даний час. Ця мова також відома як найкраща мова для створення AI та машинного навчання на основі веб-додатків. Ця мова є простою у вивченні, часто використовується для написання веб-сайтів.

Популярні веб-сайти, що зараз використовують Python: Facebook, Microsoft, Dropbox, Mozilla, Netflix, Youtube та інші.[7]

Переваги Python:

- Приваблива для швидкої розробки програм, завдяки наяності структур даних високого рівня.
- Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах.

- Підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду.
- Підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.[8]

Наступна мова програмування - Java. Java вважається найбільш стабільною мовою, і вона вижила на піку в галузі програмування з 20 років тому. Успіх ця мова отримала завдаки принципу кросплатформенності, банатопоковості та динамічному ропілу пям'яті.

Технологія Java – це і мова програмування, і платформа. На Java програмуючи, вихідний код спочатку записується як текстові файли, що пишутся та читаються людиною з розширенням .java. Вони складаються в прочитаний машиною файл .class за допомогою компілятора javac. Потім інтерпретатор Java може виконати програму з екземпляром віртуальної машини Java (Java VM).

Оскільки Java VM доступна у багатьох різних операційних системах, той самий .class файли здатні працювати в операційних системах Windows, Linux та Mac – так Java-програмісти теоретично користуються міжплатформою здатністю «писати один раз, бігайте куди завгодно ».Щоб створити програми Java, бібліотеки класів Java та компілятор javac потрібно встановити на комп'ютері. Для того, щоб запускати програми Java, Java Для постачання інтерпретатора Java потрібно встановити середовище виконання (JRE).

Всі ці компоненти містяться у вільно доступній упаковці під назвою Платформа Java, комплект для розробки стандартного видання (JDK).[9]

Окрім кросплатформенності Java має ряд переваг, таких як автоматичне управління пам'ятю, що також досягається завдяки роботі вітруально машини, багатопотоковість, що дозволяє запускати декілька паралельних процесів одночано. Також ця мова дуже часто використовується для написання веб-додатків за допомогою Servlet API.

Java Servlet API — стандартизований API для створення динамічного контентдо веб-сервера, використовуючи платформу Java. Сервлети — аналог

технологій PHP, CGI і ASP.NET. Сервлет може зберігати інформацію між багатьма транзакціями, використовуючи HTTP кукіз, сесії або через редагування URL. [10]

Servlet API, що міститься в пакеті javax.servlet, описує взаємодію веб-контейнера і сервлета. Веб-контейнер — це компонент веб-сервера, що створений для взаємодії з сервлетами. Він відповідає за управління життєвим циклом сервлетів, перетворення URL у певний сервлет та забезпечення того, щоб клієнт, який зробив URL запит, мав відповідні права доступу.[10]

Веб-сайти, що використовують Java: ebay.com, linkedin.com, aws.amazon.com, aliexpress.com, bitbucket.org, ebay.co.uk. [7]

JavaScript - це мова програмування "frontend". JavaScript широко використовується для розробки інтерактивних додатків для фронтенів. У наші дні багато організацій, зокрема стартапи, використовують NodeJS, що є середовищем виконання JavaScript. Node.js дозволяє розробникам використовувати JavaScript для сценаріїв на стороні сервера - запуск сценаріїв на стороні сервера для створення динамічного контенту веб-сторінки до відправки сторінки у веб-браузер користувача.

Плюси:

- JavaScript на стороні клієнта дуже швидкий. Він запускається негайно веб-браузері, оскільки немає вимоги до компіляції
- Надає багатший інтерфейс веб-сайту
- Це мова програмування в Інтернеті
- Зменшення запитів на сервер веб-сайтів завдяки клієнту

Мінуси:

- Відсутність копії або аналогічного методу
- Дозволяє лише одинакове успадкування
- На стороні клієнта може бути використаний для взламування веб-додатку.
- Різні браузери можуть по різномі інтерпретувати скрипти [11]

На основі вишеописаних переваг та недоліків мов програмування, для реалізації інформаційної веб-системи для онлайн запису на консультації доцільно обрати мову програмування Java. Ця мова є стабільною на широко використовується у багатьох сферах програмувааня протягом останніх 20 років. На основі Java платформи розроблені зручні API та фреймворки для веб-розробки.

3.2 Вибір фреймворку для розробки веб-додатку

У сучасному веб-програмуванні дуже часто викостовуються фреймворки. Фреймвок – це платформа для розробки програмних додатків. Він забезпечує основу, на якій розробники програмного забезпечення можуть будувати програми для певної платформи. Наприклад, фреймворк може включати заздалегідь визначені класи та функції, які можна використовувати для обробки вводу, керування апаратними пристроями та взаємодії із системним програмним забезпеченням. Це впорядковує процес розробки, оскільки програмістам не потрібно кожен раз, коли вони розробляють нову програму розробляти велику кількість базового функціоналу, що розроблений в рамках фреймворку.

Під час проектування архітектури було вирішено використовувати паттерн Model-View-Controller. Отже, для розробки інформаційної веб-системи можемо обрати сучасний фреймвокр Spring MVC.

Spring MVC – це оригінальний веб-фреймворк подудований на Servlet API і включений у Spring Framework з самого початку. Цей фреймворк є универсальним для Java платформи. Він дозволить взаємодіяти з клієнтами за допомогою принципу запит-відповідь.

Фреймворк Spring Model-View-Controller (MVC) розроблений навколо DispatcherServlet, який розсилає запити до обробників, з налаштовуваними відображеннями оброблювача, роздільною здатністю перегляду, локальним розташуванням та роздільною здатністю теми, а також підтримкою для завантаження файлів. Обробник за замовчуванням базується на анотаціях @Controller та @RequestMapping, пропонуючи широкий спектр гнучких

методів обробки. З впровадженням Spring 3.0 механізм @Controller також дозволяє створювати RESTful веб-сайти та програми через анотацію @PathVariable та інші функції.

У Spring Web MVC ви можете використовувати будь-який об’єкт як команду або об'єкт, що підтримує форму; вам не потрібно реалізовувати рамковий інтерфейс або базовий клас. Прив'язка даних Spring дуже гнучка: наприклад, вона розглядає невідповідність типу як помилки перевірки, які можна оцінити додатком, а не як системні помилки. Таким чином, вам не потрібно дублювати властивості бізнес-об’єктів як прості, нетипізовані рядки у ваших об'єктах форми, щоб просто обробити недійсні подання або належним чином перетворити рядки. Натомість часто бажано прив’язувати безпосередньо до об'єктів вашої бізнес-моделі.

Роздільна здатність перегляду весни надзвичайно гнучка. Реалізація контролера навіть може записувати безпосередньо в потік відповідей. Зазвичай екземпляр ModelAndView складається з імені перегляду та моделі моделі, яка містить назви квасолі та відповідні об'єкти, такі як команда або форма, які містять довідкові дані. Дозвіл перегляду імен можна налаштувати через назви файлів, файл властивостей або власну реалізацію ViewResolver. Модель (M у MVC) базується на інтерфейсі Map, який дозволяє здійснити повну абстракцію технології перегляду. Ви можете безпосередньо інтегрувати JSP, Velocity або будь-яку іншу технологію візуалізації. Карта моделі просто трансформується у відповідний формат, такий як атрибути запиту JSP або модель шаблону швидкості.[12]

Даний фреймвок повністю підходить для створення веб-додатку інформаційної системи для онлайн запису на консультації. Spring MVC створений на остнові кросплатфоременної мови програмування Java, завдяки йому можна логічно розділити код проекту на три різні за призначенням частини, що зробить проект гнучким до змін та майбутніх розширень системи. Фреймворк дозволить побудувати клієнт-серверну архітекруту, серверна частина веб-додатку буде швидко відповідати на запити клієнтів. Spring MVC візьме на себе функцію створення більшості об’єктів у системі та зв’язків між

ними. Цу дозволить змешити кількість коду в проекті Таким чином даний фреймворк стоворюватиме каркас майбутнього веб-додатку, а нам лише потрібно буде налаштувати нашу функціональність.

3.3 Вибір системи управління базаю даних

Інформаційна система для онлайн запису на консультації буде оперувати великою кількістю даних про користувачів, їх проведені та заплановані консультації. Зазвичай великі обсяги даних зберігаются та оброблюються базою даних.

База даних — це певний набір даних, які пов'язані між собою спільною ознакою або властивістю, та впорядковані, наприклад, за алфавітом.База даних (БД) — це організована структура, яка призначена для зберігання, зміни та обробки взаємозалежної інформації, переважно великих обсягів.БД використовують для динамічних сайтів з великими обсягами (інтернет-магазин, портал, корпоративний сайт).

Головною перевагою БД є швидкість внесення та використання потрібної інформації. Завдяки спеціальним алгоритмам, які використовуються для баз даних, можна легко знаходити необхідні дані всього за декілька секунд. Бази даних для сайтів дають змогу зберігати інформацію, що виглядає як зв'язані між собою таблиці. Саме в БД зберігаються вся необхідна та корисна інформація для функціонування сайту. [13]

Для керування БД, що зберігає дані в таблицях використовуют мову струткурованих запитів SQL.

SQL — декларативна мова програмування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними БД, створення схеми бази даних та її модифікації, системи контролю за доступом до бази даних. [14]

Це збереження та керування таким великим об’ємом даних потрібно обрати одну з сучасних систум управління базами даних.

Систе́ма управління базами даних (СУБД)— набір взаємопов'язаних даних (база даних) і програм для доступу до цих даних. Надає можливості

створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.[15]

Одніє з сучасних та широковикористовуваних систем управління базами даних є реляційна СУБД Oracle. Це система створена для керування великими обсягами даних, дає можливість розробникам створити ієрархічну структура бази, що є ефективним способом керування даними багатьох системних об’єктів, підтримує виконання ієрархічних запитів.

База даних Oracle підключається до всіх популярних мов програмування та середовищ розробників, включаючи Java, .NET, Python, Node.js, Go, PHP, C / C ++ та багато іншого. Також має безкоштовні інструменти розробки та IDE від Oracle, включаючи SQL Developer, SQLcl та SQL Developer Data Modeler. Oracle Database добре підходить для великих і маленьких користувачів.[16]

Oracle підходить для розробки додатків на основі Spring MVC фреймворку. Фреймворк має інструметри для роботи з базами даних. Отже, використання фреймовку на цієї СУБД буде доцільним при розробці додатку, оскільки це зменшить час розробки, дозволить ефективно оброблювати великі об’єми даних користувачів іформаційної системи.

4 Проектування додатку

Першим кроком у розробці будь-якої програми є створення різних допоміжних діаграм, які спрощують весь процес розробки. Для створення дизайну додатку використаємо загально прийняту мову моделювання UML.

Уніфікована мова моделювання (UML) – це сімейство графічних нотацій, в основі яких лежить єдина метамодель. [5] Вона допомагає в описанні та проектуванні інформаційних систем, особливо систем побудованих з використанням об'єктно-орієнтованих властивостей.

Для зображення можливих дій клієнт або сервера в системі використаємо use case діаграму. Суть даної діаграми полягає в наступному: проектована система подається у вигляді множини сутностей або акторів, що взаємодіють з системою за допомогою за допомогою так званих варіантів використання(use case). При цьому актором називають будь-яку зовнішню сутність, що взаємодіє з системою. Це може бути людина, технічний пристрій, програма або будь-яка інша система, яка може служити джерелом впливу на створювану систему так, як визначить сам розробник. У свою чергу, варіант використання (use case) служить для опису сервісів, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який чинять системою при діалозі з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів з системою.[6]

Даний вид діаграм використовується для зображення відношення між акторами та прецедентами(варіантами дій) в системі.

На Рисунку №4.1 зображено use case діаграму інформаційної системи.

Акторами системи являються:

- server – сервер інформаційної системи;
- web client – не зареєстрований клієнт;
- client – авторизований клієнт;
- staffer – авторизований клієнт, що є працівником клініки.

Сервер прослуховує запити клієнтів, контролює процес обміном даними, надаючи клієнтам необхідну інформацію. Клієнт, в залежності від

свого статусу, може посилати відповідні запити на сервер та отримувати від нього відповіді.

Клієнти можуть виконувати такі дії, як :

- зареєструватись;
- авторизуватись;
- переглянути можливі варіанти зустрічей;
- створити зустріч;
- редагувати існуючі заплановані зустрічі;
- переглядати графік своїх зустрічей;
- редагувати свій графік в залежності від свого статусу в системі.

Сервер може виконувати такі дії, як :

- показати список працівників та послуг;
- зареєструвати нового користувача;
- авторизувати користувача;
- сотворити зустріч;
- редагувати зустріч;
- зберегти оновлення графіку;
- повідомити користувача незадовго до запланованої зустрічі.

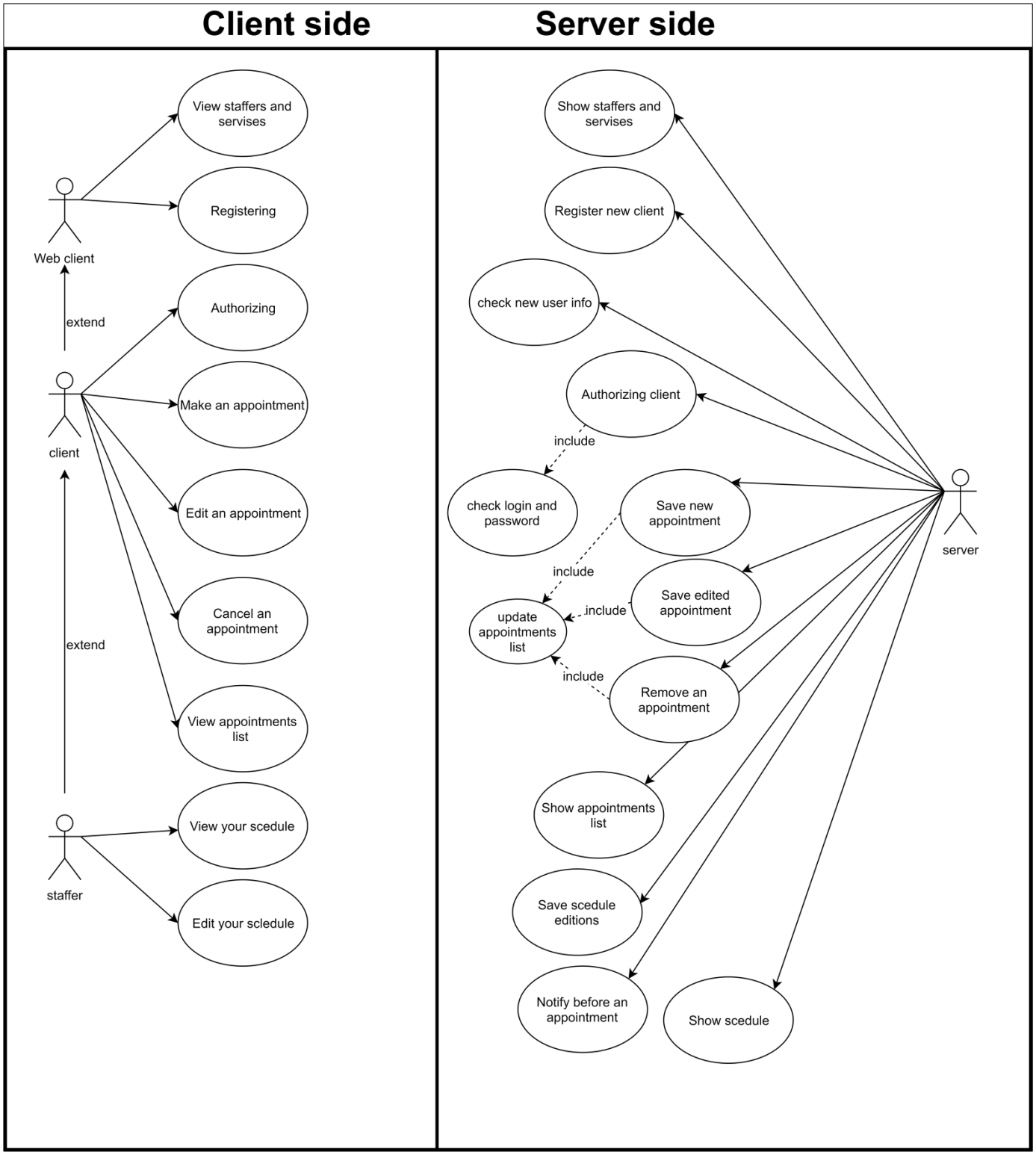

Рисунок 4.1 «Use Case Diagram»

При розробці програмного забезпечення діаграма класів в уніфікованій мові моделювання (UML) - це тип статичної структурної схеми, що описує структуру системи шляхом показу класів системи, їх атрибутів, операцій (або методів) та співвідношень між об'єктами.[5]

Діаграма класів є основним будівельним блоком об'єктно-орієнтованого моделювання. Він використовується як для загального концептуального моделювання систематики програми, так і для детального моделювання, що переводить модель в код програми. Діаграми класів також можуть бути використані для моделювання даних. Класи в діаграмі класів представляють як основні елементи, що взаємодіють в додатку, так і класи, які потрібно запрограмувати.[6]

На Рисунку №4.2 зображено діаграму класів серверної частини, на Складові діаграми побудовані згідно вище обраного патерну Model-View-Controller.

Класи сервера:

- Model:
    - User – зберігає інформацію про користувача;
    - Staffer – зберігає інформацію про працівника, розширює клас User;
    - Appointment – зберігає інформацію про зустріч;
    - DAO – інтерфейс для взаємодії з базою даних на рівні класу, використовується для визначення можливих запитів до бази даних з серверу за допомогою класу ServerController;
    - OracleDAO – клас, що реалізує інтерфейс DAO для СУБД Oracle;
- Controller:
    - ServerController – приймає запити від клієнтів, оброблює їх використовуючи методи моделі, відсилає клієнту результат;
    - ClientParser – допоміжний клас, що використовує контролер сервера для перетворення запитів від клієнта у зрозумілий для сервера формат;

- View:
    - ServerLogger – використовується для створення журналу подій на сервері.

Для інформаційної системи онлайн запису було створено діаграми варіантів використання та діаграми класів, окремо для клієнтської і серверної частини. Для створення UML діаграм було використано онлайн ресурс draw.io.

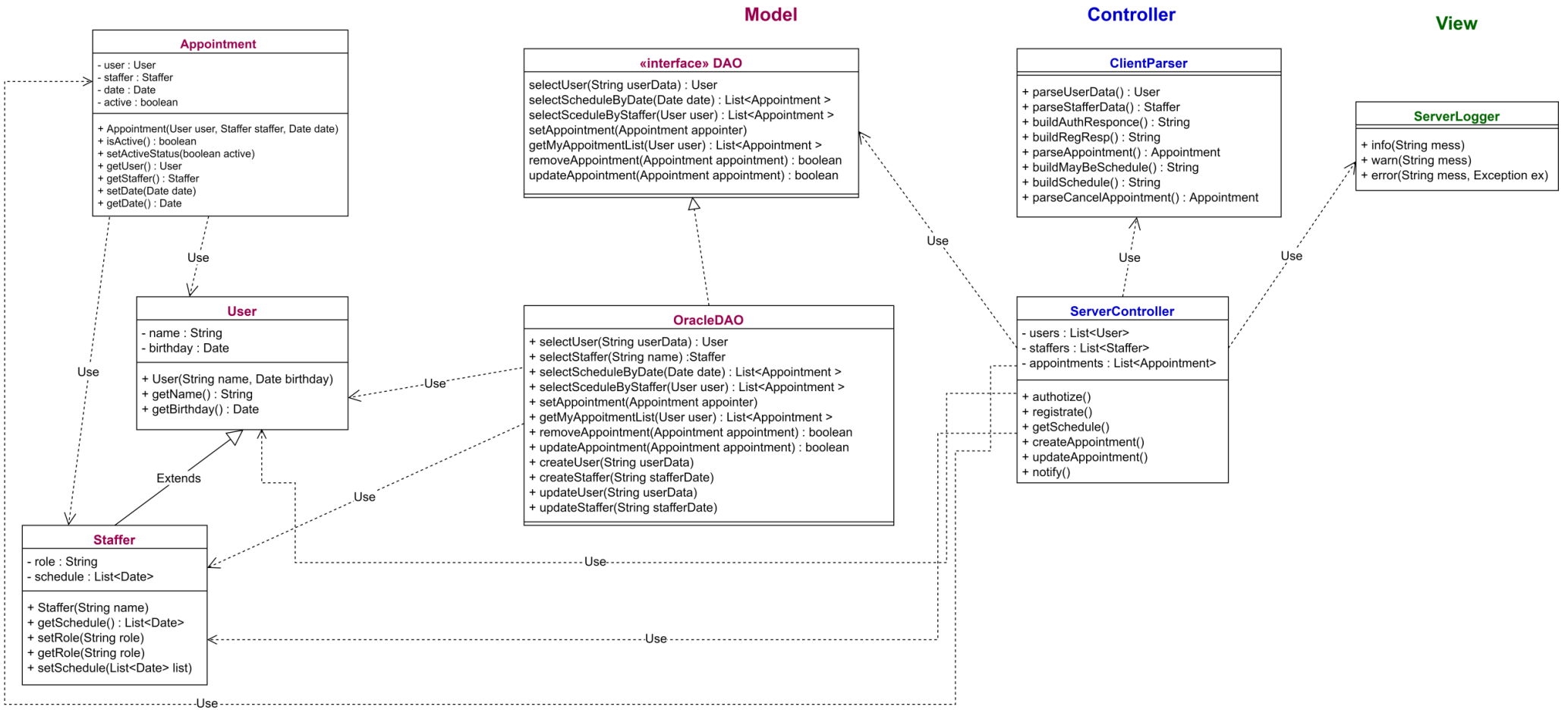

Рисунок 4.2 «Class Diagram»

5 Розробка веб-додатку для онлайн запису на консультацію

Процес розробки веб-додатку для інформаційної системи онлайн запису на консультації можна розділити на такі етапи:

- Створення бази даних та реалізація основних запитів для роботи в системі.
- Розробка клієнтської та серверної частини додатку, за допомогою Spring MVC фреймворку.

Зобразимо за помогою діаграм основні потоки даних в системі, на основі яких створимо концептуальну модель даних.

5.1 Створення бази даних

Створювана інформаційни система онлайн запису на консультації керує даними користувачі та працівників. Основні інформаційні процеси системи можна зобразити у вигляді Data Flow Diagram(DFD) – діаграма потоків даних.

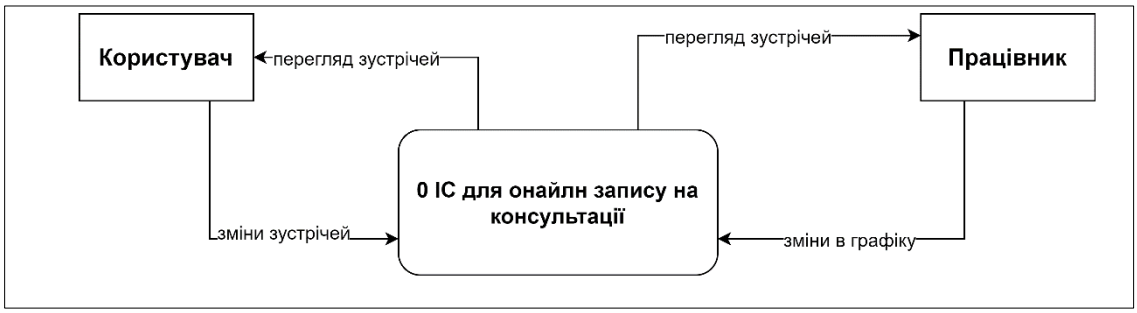

Рисунок 5.1.1 «DFD 0-го рівня ІС для онлайн запису на консультації»

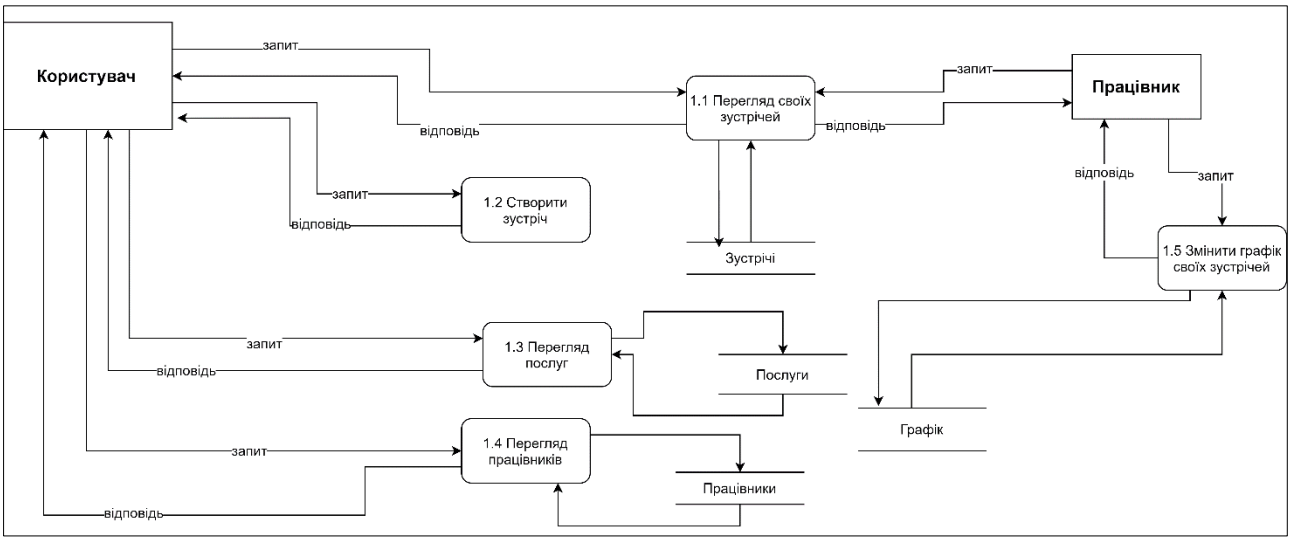

Рисунок 5.1.2 «DFD 1-го рівня ІС для онлайн запису на консультації»

Для обробки процесів концептуальна модель має наступні сховища даних: «Зустручі», «Послуги», «Графік», «Працівники». Усі запити від користувача ділятся на процеси:

- «Перегляд своїх зустрічей» - надає користувачу інформацію про всі заплановані та вже проведені зустрічі.
- «Створити зустріч» - оброблює запит від користувача на створення зустрічі.
- «Перегляд послуг» - надає користувачц інформацію про всі можливі послуги.
- «Перегляд працівників» - надає користувачу інформацію про всіх працівників, до яких можна записатися на консультацію.

Усі процеси, у яких задіяний працівник:

- «Перегляд своїх зустрічей» - надає працівнику інформацію щодо запланованих з ним консультацій.
- «Змінити графік зустрічей» - оброблює запит працівника щодо зміни його робочого графіку.

Процеси отримують запити від користувачів системи, оброблють їх та формують на них відповідь.

Зобразимо отримані зв’язки в системі за допомогою ERD.

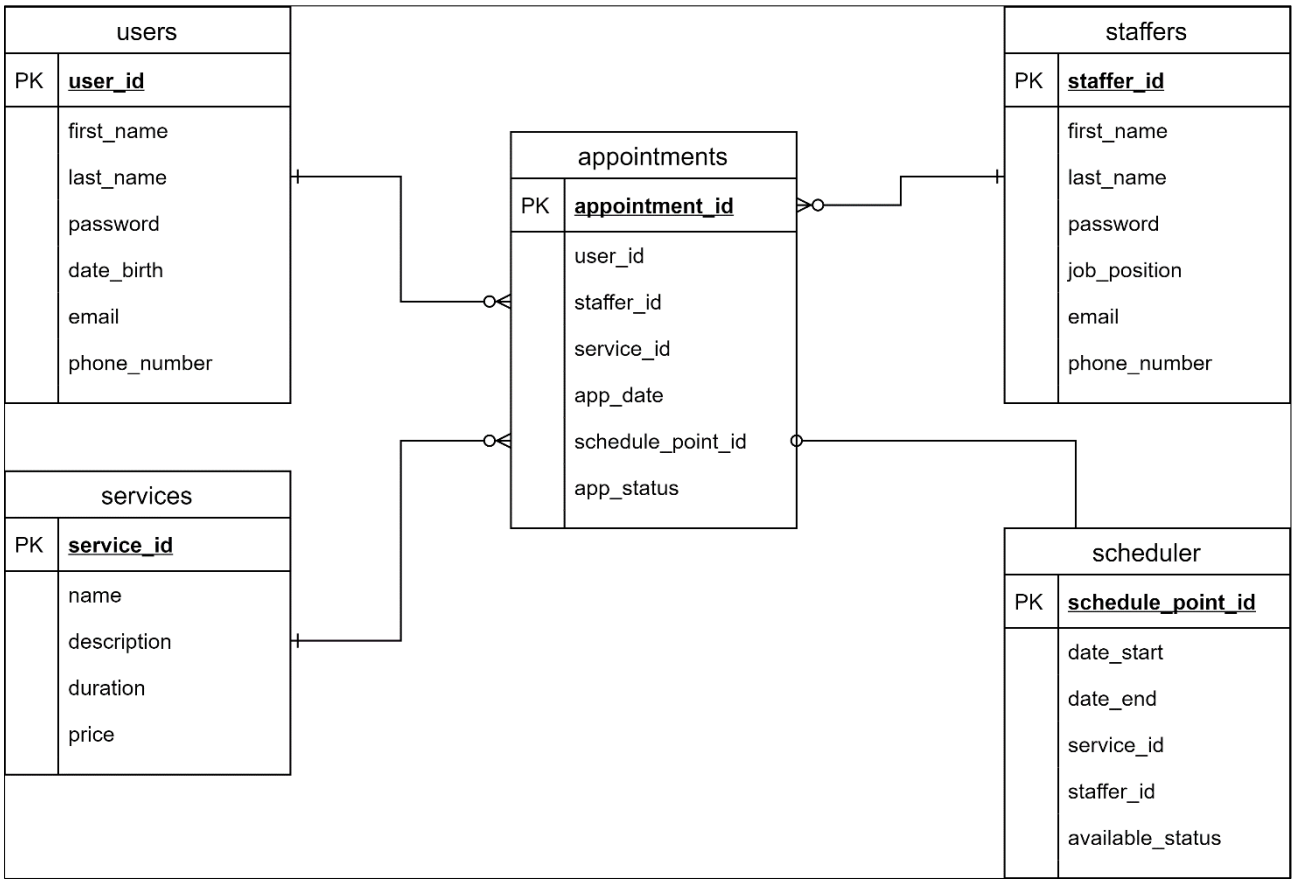

Рисунок 5.1.3 «ERD ІС для онлайн запису на консультації»

Проаналізувавши сутність, використовувані в моделі ІС, перейдемо до реалізації структури БД. Для цього представимо імена необхідних таблиць, атрибутів, типів, їх призначення та обмеження.

Окрім зображених на діаграммі сутностей, що являть собою таблиці, база даних містить послідовності для автоматично створення індентифікаторів сутностей, реалізація яких показана в додатку А.

Таблиця 5.1.1 «Детальний опис сутностей ІС»

Таблиця	**Поле**	**Зміст**	**Тип**	**Ключі**	**Обмеження**
users	user_id	Ідентифікатор користувача	NUMBER(10)	PK	Не пустий
	first_name	Ім’я користувача	VARCHAR(50)		Не пустий
	last_name	Прізвище користувача	VARCHAR(50)		Не пустий
	password	Пароль користувача	VARCHAR(15)		Не пустий
	date_birth	Дата народженя користувача	DATE		Не пустий
	email	Адреса електронної пошти	VARCHAR(20)		Не пустий
	phone_number	Номер телефону	VARCHAR(15)		Не пустий
staffers	staffer_id	Ідентифікатор працівника	NUMBER(10)	PK	Не пустий
	first_name	Ідентифікатор банку	VARCHAR(50)	PFK	Не пустий
	last_name	Курс покупки долара	VARCHAR(50)		Не пустий
	password	Курс продажу долара	VARCHAR(15)		Не пустий
	job_position	Робоча позиція працівника	VARCHAR(20)		Не пустий
	email	Адеса електронної пошти	VARCHAR(20)		Не пустий
	phone_number	Номер телефону	VARCHAR(15)		Не пустий
services	service_id	Ідентифікатор послуги	NUMBER(10)	PK	Не пустий
	name	Назва послуги	VARCHAR(50)		Не пустий
	description	Опис послуги	VARCHAR(50)		Не пустий
	duration	Час продовження	NUMBER(10)		Не пустий
	price	Ціна	NUMBER(5)		Не пустий
scheduler	schedule_poin_id	Ідентифікатор пункту графіка	NUMBER(10)	PK	Не пустий
	date_start	Час початку консультації	DATE		Не пустий

	date_end	Час закінчення консультації	DATE		Не пустий
	service_id	Ідентифікатор послуги	NUMBER(10)	PFK	Не пустий
	staffer_id	Ідентифікатор працівника	NUMBER(10)	PFK	Не пустий
	available_status	Статус зайнятості пункту графіка	VARCHAR(10)		Не пустий
appointments	appointment_id	Ідентифікатор консультації	NUMBER(10)	PK	Не пустий
	user_id	Ідентифікатор користувача	NUMBER(10)	PFK	Не пустий
	staffer_id	Ідентифікатор працівника	NUMBER(10)	PFK	Не пустий
	service_id	Ідентифікатор послуги	NUMBER(10)	PFK	Не пустий
	app_date	Дата консультації	DATE		Не пустий
	schedule_point_id	Ідентифікатор зустрічі по графіку	NUMBER(10)	PFK	Не пустий
	app_status	Статус проведення зустрічі	VARCHAR(10)		Не пустий

5.2 Розробка компонетів веб-додатку

Для розробки веб-додатку було вирішено використати мову програмування Java та Spring MVC фреймворк. Такий вибір обумовлений кросплатформенністю мови програмування Java, її популярність на широкими можливостями для створення веб-додатків.

Оскільки наш додаток містить в собі багато класів, для обробки клієнтіских запитів використовує фреймворк та має залежності від сторонніх бібліотек, то для автоматизації роботи з проектом було використано систему збірки проекту Apache Maven.

Для опису програмного проекту, який потрібно побудувати (*build*), Maven використовує конструкцію відому як Project Object Model (POM), залежності від зовнішніх модулів, компонентів та порядку побудови. Виконання певних, чітко визначених задач — таких, як компіляція коду та пакетування відбувається шляхом досягнення заздалегідь визначених цілей (targets).[17]

Build-налаштування проекту містятся в файлі pom.xml, фрегмент якого можна побачити на рисунку №5.2.1 .

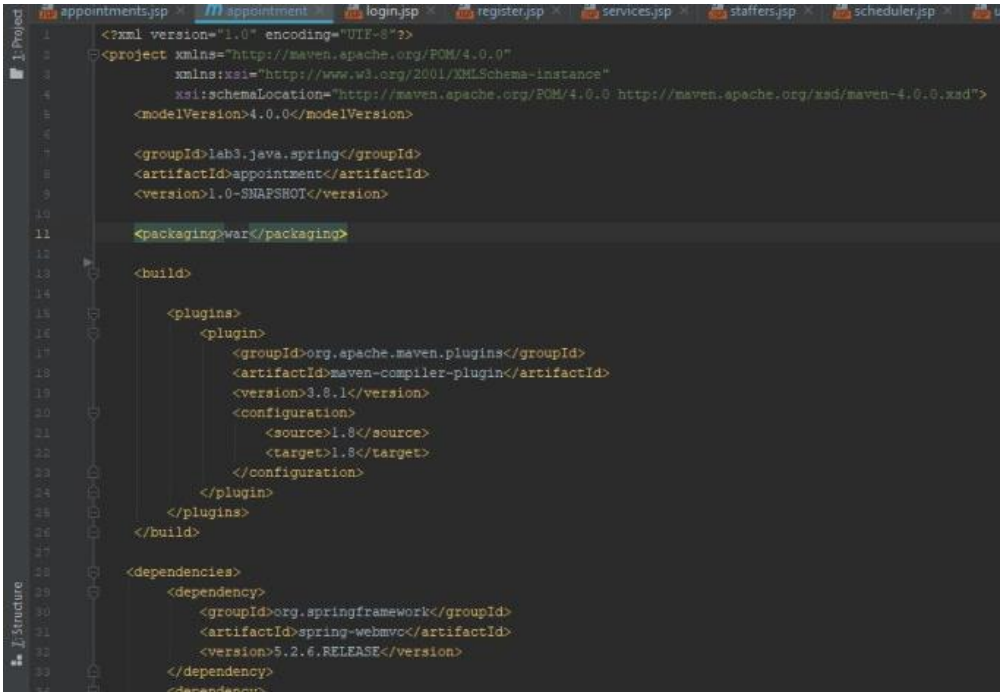

Рисунок 5.2.1 «Фрагмент файлу pom.xml»

В результаті збірки проекту мавеном формується веб-архів типу .war. Файли даного типу описують повністю запакований веб-додаток та запусткаются веб-сервером.

При розробці проекту для запуску та тестування додатку було використано веб-контейнер Apache Tomcat.

Програмне забезпечення Apache Tomcat – це реалізація з відкритим кодом технологій Java Servlet, JavaServer Pages, Java Expression Language та Java WebSocket. Java сервлет, сторінки JavaServer, мова виразів Java та специфікації Java WebSocket розробляються в рамках процесу спільноти Java. [18]

Даний веб-контейнер, на відміну від інших, є простим у використанні та не потребує багато ресурсів, що є значною перевагою під час його використання у процесі розробки.

Інформаційна система реалізованя за допомогою Spring MVC фреймворку, який для підключення до бази даних Oracle використовує безкоштовний драйвер ojbdc8.

Даний драйвер реалізує підключення до бази даних, що в свою чергу дозволяє додатку послати запити та отримувати на них відповіді від бази даних інформаційної системи.

На Рисунку №5.3.2 зображено підключення до бази даних за допомогою драйверу

```
public void connect() throws SQLException {
    try {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        connection = DriverManager.getConnection( url: "jdbc:oracle:thin:@localhost:1521:XE", user: "SYSTEM", password: "system");
        if ( connection.isClosed()) {
            LOGGER.error("connection to db is failed");
            throw new SQLException("connection failed !!!");
        }
    }
    catch (Exception ex) {
        LOGGER.error("error while connecting to data base");
        throw new SQLException(ex.getMessage(), "can't connect to db");
    }
}
```

Рисунок 5.2.2 «Підключення до бази даних»

Усі методи, що переносят дані між базою та користувачем системи реалізовані на основі спроектованої діаграми класів: інтерфейс Dao та його реалізація для OracleDao.

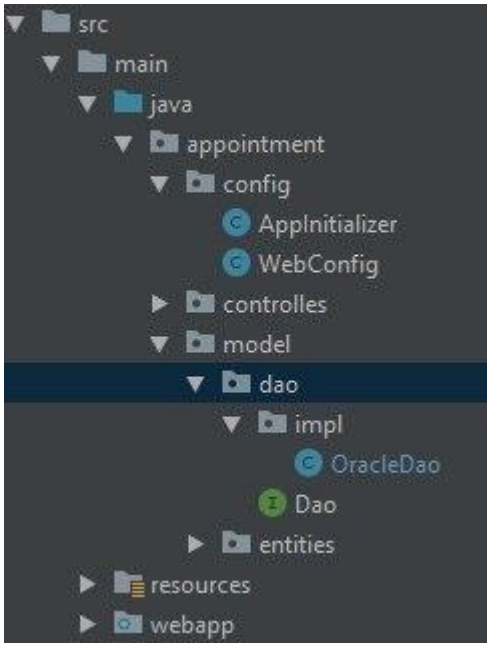

Рисунок 5.2.3 «Розташування директорій в кореневому каталозі проекту»

Кореневий каталог проекту містить такі папки:

- java – містить пакети config, controllers, model, у який знаходятся класи для конфігурації проекту, контролери для обробки клієнтських запитів, модель дотупу до проектних даних відповідно.
- webapp – містить в собі клієтську чатину веб-додатку, що реалізована за допомогою технології JSP та стандартної бібліотеки тегів JSTL.

JSP (JavaServer Pages) - технологія, що дозволяє веб-розробникам створювати вміст, який має як статичні, так і динамічні компоненти. Сторінка JSP містить текст двох типів: статичні вихідні дані, які можуть бути оформлені в одному з текстових форматів HTML, SVG, WML, або XML, і JSP-елементи, які конструюють динамічний вміст.[19]

Стандартна бібліотека тегів JSTL (англ. JavaServer Pages Standard Tag Library, JSTL) – розширення специфікації JSP, що додає бібліотеку JSP тегів для

загальних потреб, таких як розбір XML даних, умовна обробка, створення циклів і підтримка інтернаціоналізації.[20]

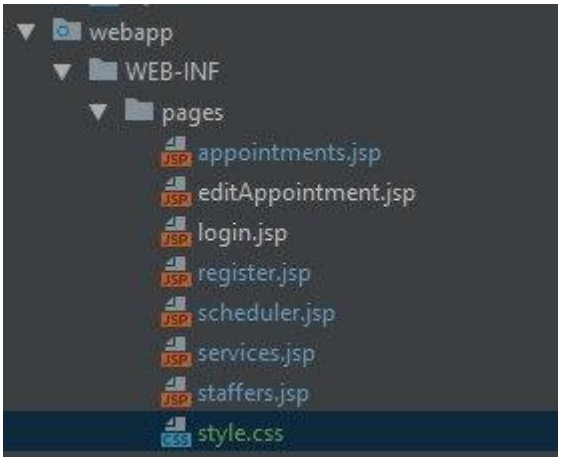

Рисунок 5.2.4 «Структура каталогу клієнтської частини веб-додатку»

Запити до бази ралізовані в класі OracleDao, який імплементує інтерфейс Dao для СУБД Oracle. Код проектних класів та опис типових запитів до бази даних можна переглянути в додатку А.

5.3 Результати реалізації веб-додатку

Для онлайн запису на консультацію, користувач в першу чергу має авторизуватися в системі. Для авторизації працівника в системі потрібно перейти за посиланням «Login as Staffer», де працівник може авторизуватись. У подальшій роботі системи користувач та працінкив працють з однаоквими сторінками додатку, відмини між їх можливостями полягають в типі авторизації.

Рисунок 5.3.1 «Сторінка авторизації»

Якщо користувач раніше не використотовува дану систему, то він може перейти за посилання для реєстрації.

Рисунок 5.3.2«Сторінка реєстрації»

Після успішної авторизації перед клієнтом системи відкривається сторінки, на якій він може побачити усі свої консультації та при бажанні змінити їх.

Рисунок 5.3.3 «Головна сторінка з усіма консультаціями користувача»

Після переходу за посиланнями для перегляду усіх послуг або працініків клієнт може на основі отриманої обрати та створити для себе консультацію.

Рисунок 5.3.4 «Сторінка усіх можливих послуг»

Рисунок 5.3.5 «Сторінка для перегляду усіх працівників»

Після створення нової консультації буде показано повідомлення про те, що консультація успішно запланованя та клієнта буде перенаправлено на головну сторінку з усіма його консультаціями.

Розроблений додаток відповідає раніше поставленим критеріям та дає можливість своїм користувача планувати свій графік за допомогою онлайн запису на консультацію до лікаря.

6 Шляхи подальшого розвитку інформаційної системи

Виділимо подальші напрямки розвитку інформаційної системи для онлайн запису на консультації.

Одним з провідних напрямків розвитку буде налагодження комунікації між користувачем на іфнормаційною системою:

- отримання звотнього зв’язку від користувача після проходження ним консультації,
- повідомлення для користувача за кілька годин перед консультацією.

Існує кілька шляхів реалізації даних критеріїв, одним з яких є використання Rest Api відправки СМС повідомлень користувачу незадовго до запланованої консультації, а також відправки на електронну пошту користувача форми з питаннями щодо якості сервісу системи та проведеної консультації.

Дані розширення допоможуть як і в налагодженні та виправленні можливих помилок в інформаційній системі, так і в налагоженні та розвитку медичного центру, що буде її використовувати.

Також планується додати можливість користувачам писати відгуки пропрацівників медичного закладу за допомогою коментарів. Для подібного розширення буде необхідно створити в базі даних сутність для збереження відгуків про працівників медичного закладу.

Якщо розглядати напрямок розвитку сімейної медицини, то буде актуальним розробка функціоналу, що дозволятиме батькам заплановувати медичні консультації для свої дітей та конторолювати їх через свій акаунт. Розробка даного функціоналу також привенесе з собою зміни в структурі сутностей базии даних.

Подяльший розвиток інформаційної системи для онлайн запису на консультації зробить можливим покращення обсуговування медичного закладу.

Висновки

Актуальність дипломного проеткту полягає в тому, що користувача мединому центру необхідна інформаціфйна система для онлайн запису на консультацію до лікаря.

Основна ідея проекту полягала в створенні зручної автоматизованої інформаційної системи, що допоможе оптимізувати час її користувачам – клієнтам клініки та медичному персоналу.

В процесі виконання дипломного проекту було проганалізовано аналогічні системи та створено список критеріїв, яким має відповідати інформаційна система. В результаті огляду подібних інформаційних систем було вирішено створити власну інформаційну систему, оскільки оглянуті системи мають як переваги так і недоліки.

Створено список вимог, яким має відповідати майбутня інформаційна система, серед яких: запис на консультацію, можливість її відміни або редагування, перегляд можливих варіантів консультацій.

Для проектування інформаційної системи було обрано клієнт-серверну архітектурну модель, оскільки дана модель зможе забезпечити швидку комунікацію між користувачем та працівниками клініки. За допомогою такого підходу система буде структурована, легко керована та матиме можливість швидкого внесення змін.

Було оглянуто та обрано патерни проектування окремо для клієнтської та серверної частини системи. Спроектовані діаграмми класів та варінтів використання, діаграма відображення потоків даних використані в процесі розробки.

Під час розробки проету було використано мову прогрмування Java, Spring MVC фреймворк, для реалізації бази даних використано СУБД Oracle. Spring MVC фреймвокр створений на основі Java платформи, ідеально підійшов для створення нашої інформаційної веб системи, оскільки дозволив реалізувати

клієнт-серверну архітектуру та дав можливість створити проект на основі патерну MVC.

Основні потоки дані в інформаційній системі показано за допомого діанрами потоків даних, структуру бази даних відображає діаграма сутностей та зв’язків. В процесі роботи додаток звертається до бази даних через інтерфейс Dao.

Для збірки та запуску проекту в процесі розробки використано Apache Maven та контейнер для веб-додатків Apache Tomcat.

В результаті виконання дипломного проекту маємо веб-додаток інформаційної системи для онлан запису на консультацію до лікаря. За допомогою додатку користувач може переглядати всі свої зустрічі, переглядати можливі послуги медичного центру та його працівників, на основі отриманої інформації запланувати для себе нові необхідні консультації. Працівник системи може переглядати заплоновані зустрічі та редагувати графік роботи. Розроблена система відповідає раніше поставленим критеріям.

Список використаної літератури

1. Что такое архитектура программного обеспечения [Електронний ресурс] – 2015. – Режим доступу: ibm.com/developerworks/ru/library/eeles/index.html – Назва з екрану.
2. Fielding, R.T. Dissertation Architectural Styles and the Design of Network-based Software Architectures. – 20 p.
3. Saternos, C. Client-Server Web Apps with JavaScript and Java. – 12 c.
4. Швец, А. Погружение в паттерны проектирования. – 5 с.
5. Tsonev, Kh. React in Patterns. . – 13 p.
6. Head First Design Pattern Eric Freeman, Elisabeth Robson, Bert Bates, Kathy Sierra – O'Reilly Media, 2004. – 694 p.
7. BEST PROGRAMMING LANGUAGES FOR WEB DEVELOPMENT IN 2020 [Електронний ресурс] – 2020. – Режим доступу: manifera.com/best-programming-languages-for-web-development-in-2020/ - Назва з екрану.
8. Python [Електронний ресурс] – 2020. – Режим доступу: uk.wikipedia.org/wiki/Python – Назва з екрану.
9. H. Terrace, H. Walk Java in easy steps – p. 15
10. Сервлет [Електронний ресурс] – 2015 –Режим доступу uk.wikipedia.org/wiki/Сервлет – Назва з екрану.
11. 10 Best Programming Languages to Learn in 2020 [Електронний ресурс] – 2020. – Режим доступу: hackr.io/blog/best-programming-languages-to-learn-2020-jobs-future – Назва з екрану.
12. Spring [Електронний ресурс] – 2020. – Режим доступу: docs.spring.io/spring/docs/3.0.x/spring-framework-reference/html/mvc.html – Назва з екрану.
13. Що таке база даних? [Електронний ресурс] – 2017. – Режим доступу: apeps.kpi.ua/shco-take-basa-danykh – Назва з екрану.
14. SQL [Електронний ресурс] – 2020. – Режим доступу: uk.wikipedia.org/wiki/SQL – Назва з екрану.

15. Система управління базами даних [Електронний ресурс] – 2006. – Режим доступу: uk.wikipedia.org/wiki/Система_управління_базами_даних – Назва з екрану.
16. Free Oracle Database for Everyone [Електронний ресурс] – 2018. – Режим доступу: oracle.com/database/technologies/appdev/xe.html – Назва з екрану.
17. Apache Maven [Електнонний ресурс] – 2020. – Режим доступу: uk.wikipedia.org/wiki/Apache_Maven – Назва з екрану.
18. Apache Tomcat Maven [Електнонний ресурс] – 2020. – Режим доступу: tomcat.apache.org/ – Назва з екрану.
19. JSP [Електнонний ресурс] – 2020. – Режим доступу: uk.wikipedia.org/wiki/JSP – Назва з екрану.
20. JSTL [Електнонний ресурс] – 2020. – Режим доступу: en.wikipedia.org/wiki/JavaServer_Pages_Standard_Tag_Library – Назва з екрану.

Додаток А

Скрипти для створення бази даних

```
CREATE TABLE users
( user_id number(10) NOT NULL,
  first_name varchar2(50) NOT NULL,
  last_name varchar2(50) NOT NULL,
  password varchar2(15) NOT NULL,
  date_birth DATE NOT NULL ,
  email varchar2(20) NOT NULL,
  phone_number varchar2(15) NOT NULL,
  CONSTRAINT users_pk PRIMARY KEY (user_id)
);
CREATE TABLE staffers
( staffer_id number(10) NOT NULL,
  first_name varchar2(50) NOT NULL,
  last_name varchar2(50) NOT NULL,
  password varchar2(15) NOT NULL,
  job_position varchar2(20) NOT NULL ,
  date_birth DATE NOT NULL ,
  email varchar2(20) NOT NULL,
  phone_number varchar2(15) NOT NULL,
  CONSTRAINT staffers PRIMARY KEY (staffer_id),
  mgr_id int REFERENCES staffers(staffer_id)
);
create table services (
      service_id number(10) not null,
      name varchar2(20) not null,
      description varchar2(50) not null,
      duration number(10) not null,
      price number(10) not null,
      constraint service_pk primary key (service_id)
);
create table appointments (
      appointment_id number(10) not null,
      user_id number(10) not null ,
      staffer_id number(10) not null,
      service_id number(10) not null,
      constraint appointment_pk primary key (appointment_id),
      constraint user_fk foreign key (user_id) references users(user_id),
```

```
            constraint staffer_fk foreign key (staffer_id) references
staffers(staffer_id),
            constraint service_fk foreign key (service_id) references
services(service_id)
      );

      ALTER TABLE appointments
        ADD CONSTRAINT schedule_point_id_fk
          FOREIGN KEY (schedule_point_id)
            REFERENCES SCHEDULE(SCHEDULE_POINT_ID);

create table SCHEDULE
(
  SCHEDULE_POINT_ID NUMBER(10)   not null
    constraint SCHEDULE_POINT_ID_PK
      primary key,
  DATE_START        DATE         not null,
  DATE_END          DATE         not null,
  SERVICE_ID        NUMBER(10)   not null,
  STAFFER_ID        NUMBER(10)   not null,
  AVAILABLE_STATUS  VARCHAR2(10) not null
);
      alter table SCHEDULE
      add constraint schedule_point_id_pk primary key (SCHEDULE_POINT_ID);
create sequence USER_SEQUENSE
  increment by 2
/
create sequence APP_SEQUENCE
  increment by 2
```

Реалізація типових запитів до бази даних

Запит для створення нового користувача в системі, використовується під час реєстрації:

```
      insert into users(user_id, first_name, last_name, password, date_birth,
email, phone_number) VALUES(USER_SEQUENSE.nextval, ?, ?, ?, ?,?, ?);
```

Даний запит використовує послідовність USER_SEQUENSE (дивись дод. А) для автоматичного створення ідентифікатору користувача.

Запит для створення нового працівника:

```
      insert into staffers(staffer_id, first_name, last_name, password,
job_position, date_birth, email, phone_number, mgr_id)
      VALUES (USER_SEQUENSE.nextval, ?, ? , ? , ? , ?, ? , ? , ?);
```

Запист для стоворення нового пункту в графіку працівника:

```
insert into SCHEDULE(schedule_point_id, date_start, date_end, service_id, staffer_id, available_status) VALUES(schedule_sequence.nextval, ?, ?, ?, ?, 'free');
```

Запит для створення нової консультації:

```
insert into appointments(appointment_id, user_id, staffer_id,service_id, app_date, SCHEDULE_POINT_ID)
VALUES (APP_SEQUENCE.nextval, ?, ?, ?, ?, ?);
```

Після створення нової консультації графік для працівників оновлюється, за допомогою запиту:

```
update schedule set available_status = ?where SCHEDULE_POINT_ID = ?");
```

Запит, що використовуються при авторизвції користувача:

```
select * from USERS where email = ? and PASSWORD = ?;
```

Запит, що знаходить всі проведені та не проведені консультації користувача:

```
select u.first_name, u.last_name, st.first_name, st.last_name, ser.name, app.app_date, to_char(sh.date_start, 'hh24:mi') ttime_start, sh.SCHEDULE_POINT_ID
from appointments app, users u, services ser, staffers st, schedule sh where app.user_id = u.user_id
and ser.service_id = app.service_id
and st.staffer_id = app.staffer_id
and app.schedule_point_id = sh.schedule_point_id
and app.user_id = ?
order by app.app_date desc;
```

Запит, що знаходить пункт в графіку в залежності від послуги:

```
select sh.SCHEDULE_POINT_ID, to_date(sh.date_start) ddate_start, to_char(sh.date_start, 'hh24:mi') ttime_start,
to_date(date_end) ddate_end, to_char(date_end, 'hh24:mi') ttime_end,
sh.service_id, sh.staffer_id, sh.AVAILABLE_STATUS, ser.name, st.first_name, st.last_name from SCHEDULE sh, services ser, staffers st where sh.service_id = ?
and ser.service_id = sh.service_id and st.staffer_id = sh.staffer_id;
```

Запит, що знаходить пункт в графіку в залежності від працівника:

```
select sh.SCHEDULE_POINT_ID, to_date(sh.date_start) ddate_start, to_char(sh.date_start, 'hh24:mi') ttime_start,
to_date(date_end) ddate_end, to_char(date_end, 'hh24:mi') ttime_end,
sh.service_id, sh.staffer_id, sh.AVAILABLE_STATUS, ser.name, st.first_name, st.last_name from SCHEDULE sh, services ser, staffers st where st.staffer_id = ?
and ser.service_id = sh.service_id and st.staffer_id = sh.staffer_id;
```

Код проекту

Файл web-конфігурації проекту WebConfig.java .

```
package appointment.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.ViewResolver;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;
import org.springframework.web.servlet.view.InternalResourceViewResolver;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "appointment")
public class WebConfig {

    @Bean
    ViewResolver viewResolver() {
        InternalResourceViewResolver viewResolver = new
InternalResourceViewResolver();
        viewResolver.setPrefix("WEB-INF/pages/");
        viewResolver.setSuffix(".jsp");
        return viewResolver;
    }
}
```

Файл appointment.jsp використовується.

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
```

```
<head>
    <title>Appointments</title>

</head>
<body>
<%--<c:set var ="user" value="${user}"/>--%>
<div class = "appHeader">
    <h1>Appointments</h1>
        <h2>Welcome to Appointment system !</h2>

    <p><a href="/services">All services</a></p>
    <form:form  action="/services" method="post">

        <div class="container">
            <input  type="text"  name="userId"  required  value="${user.userId}"
hidden>

            <button type="submit">All services</button>
        </div>

    </form:form>
    <p><a href="/staffers">All staffers</a></p>
    <h2>All your appointments</h2>
   <%-- <c:import url="userLink.jsp"> </c:import>--%>
</div>
    <table>
    <p>${errorListMsg}</p>
        <c:if test="${appointmentList.size() > 0}">
            <tr>
                <th>Staffer</th>
                <th>Service</th>
```

```
                <th>Date</th>
                <th>Time</th>
                <th>Action</th>
            </tr>

        <c:forEach var="film" items="${appointmentList}" varStatus="i">
            <tr>
                <td> ${film.staffer.firstName} ${film.staffer.lastName}</td>
                <td> ${film.service.name}</td>
                <td> ${film.appointmentDate}</td>
                <td> ${film.schedulePoint.timeStart}</td>
                <td>
                    <form:form  action="/delete" method="post">

                    <div class="container">
                        <input       type="text"       name="id"       required
value="${film.appointmentId}" hidden>

                        <button type="submit">Delete</button>
                    </div>

                    </form:form>
                    <form:form  action="/edit" method="post">

                        <div class="container">
                            <input      type="text"      name="id"      required
value="${film.appointmentId}" hidden>

                            <button type="submit">Edit</button>
                        </div>
```

```
                    </form:form>
                </td>
            </tr>

        </c:forEach>
        </c:if>
    </table>

    <%--<h2><a href="/edit">Edit appointment page</a></h2>--%>

<p>${appointment}</p>
</body>
</html>
```

Файл login.jsp використовується при авторизації користувача.

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Login</title>

</head>
<body>
<form:form  action="/login" method="post">

    <div class="container">
        <p> ${error}</p>
        <label><b>Email</b></label>
        <input type="email" placeholder="Enter Username" name="email" id = "email"
required>

        <label><b>Password</b></label>
```

```
        <input type="password" placeholder="Enter Password" name="password" id =
"password" required>

        <button type="submit">Login</button>
    </div>

</form:form>

<a href="/register"><p>Registration</p></a>

<a href="/stafferLogin"><p>Login as Staffer</p></a>
</body>
</html>
```

Файл register.jsp використовується для реєстрації нового користувача в системі.

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Register</title>
</head>
<body>
<form:form  action="/register" method="post">

    <div class="container">
        <p> ${error}</p>
        <label><b>Fist name</b></label>
        <input  type="text"  placeholder="Enter  Username"  name="firstName"  id =
"firstName" required>
```

```
        <label><b>Last name</b></label>
        <input type="text" placeholder="Enter Username" name="lastName" id =
"lastName" required>


        <label><b>Email</b></label>
        <input type="email" placeholder="Enter Username" name="email" id = "email"
required>

        <label><b>Password</b></label>
        <input type="password" placeholder="Enter Password" name="password" id =
"password" required>

        <button type="submit">Login</button>
    </div>

</form:form>
</body>
</html>
```

Файл services.jsp використовується для відображення користувачу усіх послуг.

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<head>
    <title>Services</title>
</head>
<body>
```

```
<h1>All services</h1>
<c:import url="userLink.jsp"> </c:import>
<table class="blueTable">
    <c:if test="${serviceList.size() > 0}">
        <tr>
            <th>Name</th>
            <th>Description</th>
            <th>Duration</th>
            <th>Price</th>
            <th>Action</th>
        </tr>
    </c:if>

    <c:forEach var="film" items="${serviceList}" varStatus="i">
        <tr>
            <td> ${film.name}</td>
            <td> ${film.description}</td>
            <td> ${film.duration} minutes</td>
            <td> ${film.price} $</td>
            <td>
                <form:form  action="/scheduler" method="post">

                    <div class="container">
                        <input   type="text"   name="id"   id   =   "id"   required
value="${film.serviceId}" hidden>
                        <input      type="text"      name="userId"      required
value="${user.userId}" hidden>
                        <button   type="submit">Create   appointment   with   with
service</button>
                    </div>
```

```
                </form:form>
            </td>
        </tr>

    </c:forEach>
</table>

</body>
</html>
```

Файл staffers.jsp використовується для відображення користувача інформації про усіх працівників.

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<html>
<head>
    <title>Staffers</title>
</head>
<body>
<h1>All Staffers</h1>
<table class="blueTable">
    <c:if test="${stafferList.size() > 0}">
        <tr>
            <th>Name</th>
            <th>Job position</th>
            <th>Email</th>
            <th>Phone Number</th>
        </tr>
    </c:if>
```

```
    <c:forEach var="film" items="${stafferList}" varStatus="i">
        <tr>
            <td> ${film.firstName} ${film.lastName}</td>
            <td> ${film.jobPosition}</td>
            <td> ${film.email}</td>
            <td> ${film.phoneNumber}</td>
            <td>
                <form:form  action="/schedulerByStaffer" method="post">

                    <div class="container">
                        <input  type="text"  name="id"  id  =  "id"  required
value="${film.stafferId}" hidden>

                        <button  type="submit">Create  appointment  with  with
service</button>
                    </div>

                </form:form>
            </td>
        </tr>

    </c:forEach>
</table>
</body>
</html>
```

Основну частину запитів від клієнта оброблює клас AppointmentController.java наведемо кілька основних методів для обробки запитів з цього класу.

Метод для оброки запиту на авторизацію користувача:

```
@RequestMapping(value = "/login", method = RequestMethod.POST)
```

```
        public ModelAndView loginUser(@RequestParam String email, @RequestParam
String password,
                                      HttpSession httpSession, Model model) {
            User user = new User(email, password);
            try {
                if(oracleDao.loginUser(user)) {
                    User appUser = oracleDao.getUserByEmail(user.getEmail());
                    List<Appointment>            appointmentList            =
oracleDao.findAppointmentsByUser(appUser);
                    ModelAndView modelAndView = new ModelAndView();
                    modelAndView.setViewName("appointments");
                    modelAndView.addObject("appointmentList",
appointmentList);
                    modelAndView.addObject("user", appUser);
                    if(appointmentList == null || appointmentList.size() == 0)
{
                        modelAndView.addObject("errorListMsg", "You don't have
any appointments now. Go to all services page for planing");
                    }
                    httpSession.setAttribute("user", appUser.getUserId());
                    return modelAndView;
                } else {
                    ModelAndView modelAndView = new ModelAndView();
                    modelAndView.setViewName("login");
                    modelAndView.addObject("error", "Login failed !");
                    return modelAndView;
                }
            } catch (SQLException e) {
                return serverError();
            } catch (ClassNotFoundException e) {
                return serverError();
            }
        }
```

Метод для обробки запиту на реєстрацію нового користувача:

```
        @RequestMapping(value = "/register", method = RequestMethod.POST)
        public   ModelAndView   register(@RequestParam   String   firstName,
@RequestParam String lastName,
                                         @RequestParam String email, @RequestParam
String password,
                                         HttpSession httpSession, Model model) {
```

```
            User user = new User(firstName, lastName, email, password);
            if(oracleDao.createUser(user)) {
                ModelAndView modelAndView = new ModelAndView();
                modelAndView.setViewName("login");
                LOGGER.debug("new user was successfully registered in the
system");
                return modelAndView;
            }
            else {
                ModelAndView modelAndView = new ModelAndView();
                modelAndView.setViewName("register");
                modelAndView.addObject("error", "Server error while registering
. . . ");
                return modelAndView;
            }
        }
```

Метод для обробки запиту на відображення інформації про всі послуги:

```
    @RequestMapping(value = "/services", method = RequestMethod.POST)
    public ModelAndView allServices(@RequestParam int userId) {
        User user = oracleDao.getUserById(userId);
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("services");
        List<Service> serviceList = oracleDao.findAllServices();
        modelAndView.addObject("serviceList", serviceList);
        modelAndView.addObject("user", user);
        return modelAndView;
    }
```

Метод для обробки запиту на відображення інформації про всіх працівників:

```
@RequestMapping(value = "/staffers", method = RequestMethod.GET)
    public ModelAndView allStaffers() {
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("staffers");
        List<Staffer> stafferList = oracleDao.findAllStaffers();
        modelAndView.addObject("stafferList", stafferList);
        return modelAndView;
    }
```

Метод для обробки запиту на створення нової консультації:

```
@RequestMapping(value = "/createApp", method = RequestMethod.POST)
    public ModelAndView createApp(@RequestParam int id, @RequestParam int
userId, HttpSession httpSession) {
        User user = oracleDao.getUserById(userId);
        SchedulePoint schedulePoint = oracleDao.getSchedulePointById(id);
        Appointment appointment = new
Appointment(user,schedulePoint.getStaffer(),
schedulePoint.getService(),schedulePoint);
        oracleDao.createAppointment(appointment);
        List<Appointment> appointmentList =
oracleDao.findAppointmentsByUser(user);
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("appointments");
        modelAndView.addObject("appointmentList", appointmentList);
        modelAndView.addObject("errorListMsg", "<script>\n" +
                "    alert( 'New Appointment was successfully created' );\n
    </script>");
        modelAndView.addObject("user", user);
        return modelAndView;
    }
```

Метод для обробки запиту на видалення існуючої консультації:

```
    @RequestMapping(value = "/delete", method = RequestMethod.POST)
    public ModelAndView deleteAppointment(@RequestParam int id) {
        Appointment appointment = oracleDao.getAppointmentById(id);
        oracleDao.deleteAppointment(appointment);
        User user =
oracleDao.getUserById(appointment.getUser().getUserId());
        List<Appointment> appointmentList =
oracleDao.findAppointmentsByUser(user);
        ModelAndView modelAndView = new ModelAndView();
        modelAndView.setViewName("appointments");
        modelAndView.addObject("appointmentList", appointmentList);
        modelAndView.addObject("errorListMsg", "<script>\n" +
                "    alert( 'Appointment was successfully deleted from your
list' );\n" +
                "  </script>");
        return modelAndView;
    }
```