ЗМІСТ

ВСТУП

На сьогоднішній день широкої популярності набрали програми, за допомогою яких користувачі в режимі реального часу можуть передавати текстові повідомлення та інший доступний контент. Такі програми звуться «месенджерами».

Їх популярність на сьогоднішній день дійсно колосальна, тому розробники мають мільярдні прибутки. Отже, месенджер – це програма для миттєвого обміну повідомленнями. Працюють такі утиліти за допомогою інтернет-з'єднання.

В ході курсової роботи було розроблено веб-додаток під назвою «xMessenger», який по своєму функціоналу відповідає мінімальним вимогам самого поняття «месенджер». Але, слід зазначити, що основною ціллю роботи є сам процес розробки ПП, оскільки він є неоднозначним і передбачає використання багатьох сучасних технологій.

Актуальність теми. Дана тема курсової роботи актуальна тим, що розробка веб-додатків займає левову частину продуктів на ІТ ринку, й уміння розбиратися у внутрішній структурі такого продукту є запорукою успішного кар'єрного росту у цій сфері.

Об'єктом дослідження є детальний процес розробки браузерного месенджера.

Мета роботи полягає у тому, щоб опанувати сучасні фреймворки й технології, закріпивши отримані вміння на практиці.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Що таке месенджер?

У перекладі з англійської месенджер – «зв'язковий» або «кур'єр». Відмінність програми від електронної пошти в тому, що повідомлення передаються дуже швидко. За допомогою E-mail так швидко розмовляти не вийде, адже опитування скриньки проводитися раз на 5-10 хвилин. Якщо порівнювати з реальним життям, то можна сказати, що електронна пошта – це листи, а месенджери – телеграми. Користувачі такі утиліти нерідко називають інтернет-пейджер, так як технологія обміну інформацією дуже схожа на простий приймач персонального виклику, але надає набагато більше можливостей.

Історія месенджерів у вигляді універсальних настільних додатків, а не систем із обмеженим доступом, починається 1996 року. Саме тоді ізраїльська компанія Mirabilis запустила ICQ. Відмінністю цієї програми стали багатокористувацькі чати, підтримка передачі файлів, пошук по базі користувачів та низка інших опцій, яких раніше не було.

Система обміну миттєвими повідомленнями працює так: коли з'являються на лінії всі абоненти, записані в телефоні або контакт-листі, вас про це сповіщають. Точно так само ваші друзі бачать, що ви перебуваєте в мережі. При цьому ви можете обмінюватися інформацією, яку видно на панельці вашого екрану і у вікні гаджета співрозмовника, практично відразу. Єдине, що потрібно для такого спілкування – однаковий месенджер з людиною, з яким ви хочете обмінюватися повідомленнями.

Список найбільш розповсюджених месенджерів станом на 2019 рік:

- ✓ *Whatsapp*
- ✓ *Viber*
- ✓ *Messenger в Facebook*
- ✓ *Skype*
- ✓ *Telegram*
- ✓ *Google Talk*
- ✓ *IMessenger*

1.2. Технічне завдання

Написання випускного кваліфікаційного проекту передбачає наступне технічне завдання:

Таблиця 1.1.

ЗАГАЛЬНІ ВІДОМОСТІ

Найменування додатку	xMessenger
Планові терміни початку та закінчення робіт	Грудень 2018 – Травень 2019
Головний бенефіціар	Відсутній. ПП не має комерційного характеру.
Потенційні користувачі	Будь-яка людина, що виявляє бажання обмінюватися інформацією електронним шляхом через мережу Інтернет.
Призначення додатку	Надати користувачеві якісний та зручний сервіс для обміну текстовими повідомленнями в режимі реального часу.
Мета створення додатку	Опанування сучасних технологій для розробки веб-додатків та їх закріплення на практиці. Набуття навичок роботи з хмарними платформами.

В ході дослідження поняття «месенджер» було сформовано перелік основних (мінімальних) та додаткових вимог до кінцевого програмного продукту. Функціональні характеристики ПП перераховані у таблиці 1.2.

Таблиця 1.2.

ФУНЦІОНАЛЬНІ ВИМОГИ ДО СИСТЕМИ

Основні	➢ реєстрація/авторизація користувача
	➢ пошук існуючих користувачів в системі
	➢ управління запитами дружби
	➢ створення/видалення чатів
	➢ обмін повідомленнями у реальному часі
	➢ видалення облікового запису
	➢ адаптація під різні пристрої
Додаткові	✓ реєстрація/авторизація користувача за допомогою сервісу Gmail (інтеграція з платформою Google)
	✓ доступ до налаштувань облікового запису
	✓ реалізація функціоналу “DnD” для видалення чатів
	✓ сповіщення на рівні ОС
	✓ налаштування теми інтерфейсу
	✓ масштабування фотокарток користувачів

Вимоги до програмного забезпечення

Програмний комплекс повинен бути побудований на клієнт-серверній архітектурі з використанням веб-технологій, що не вимагають додаткового ліцензування. Серверна частина програмного комплексу має бути реалізована програмною мовою Java, з використанням системи керування базами даних PostgreSQL на базі веб-сервера Apache. При проектуванні програмного комплексу необхідно застосувати архітектуру Model-View-Controller для розмежування алгоритмічної частини від інтерфейсу та забезпечення спільного використання компонентів.

Клієнтська частина повинна бути побудована на веб-технологіях, що дозволяють будувати багатофункціональні веб-компоненти. Сторінки інтерфейсу генеруються стандартом HTML5, оформлення здійснюється за

допомогою таблиць стилів CSS, а інтерактивність сторінок забезпечується асинхронними запитами через AJAX. Інтерфейс повинен коректно працювати у сучасних веб-браузерах Mozilla FireFox та Google Chrome.

Кінцевий ПП має бути доступним через мережу Інтернет, тому сам додаток має бути розгорнуто на хмарній платформі.

Вимоги до технічного забезпечення

ПП розрахований на функціонування при такому наборі технічних засобів:

- мінімальна апаратна конфігурація сервера:
  - 2-ядерний процесор з тактовою частотою від 1.4ГГц;
  - оперативна пам’ять об’ємом не менше 4Гб;
  - попередньо встановлені програмні модулі – NPM (5<) та JRE (8<);
- мінімальна апаратна конфігурація комп’ютера-клієнта:
  - веб-браузер Mozilla FireFox (39<) або Google Chrome (42<);
  - підключення до мережі Інтернет.

Технічне забезпечення надається хостинговою веб-платформою (ресурси передбачені самою хмарою).

Висновки до Розділу 1

Концепція месенджерів сягає своїм корінням середини 1960-х. Впродовж цього часу месенджери як ПП еволюціонували та значно розширили перелік функціональних особливостей. На сьогодні існує досить велика кількість таких програм, кожна з яких надає унікальний сервіс користувачеві.

Аналізуючи результати проведеного вище дослідження, було розроблено технічне завдання ПП та висвітлено рекомендації щодо його реалізації.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1. Середовище розробки ПП

Інтегровані середовища розробки (ICP, англ. *Integrated development environment* або англ. *IDE*) створені для того, щоб максимізувати продуктивність програміста, надавши йому пов'язані інструменти розробки зі схожими інтерфейсами як одну програму, в якій відбуватиметься весь процес розробки й яка надає необхідні функції для модифікації, компілювання, розгортання та налагодження програмного забезпечення.

Розробка ПП повністю проводилася у ICP «*IntelliJ IDEA*» – комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP та ін.) від компанії JetBrains. Система поставляється у вигляді урізаної по функціональності безкоштовної версії «Community Edition» і повнофункціональної комерційної версії «Ultimate Edition» (при написанні проекту була використана розширена версія), для якої активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію.

Рис. 2.1. Логотит ICP «IntelliJ IDEA Ultimate Version»

2.2. СУБД PostgreSQL

Для зберігання і обробки даних була обрана СУБД PostgreSQL. Її багатофункціональність та надійність були основними критеріями вибору серед інших кандидатів.

PostgreSQL — об'єктно-реляційна система управління базами даних (СУБД). Є альтернативою як комерційним СУБД (Oracle Database, Microsoft SQL Server, IBM DB2 та інші), так і СУБД з відкритим кодом (MySQL, Firebird, SQLite).

Рис. 2.2. Логотип СУБД «PostgreSQL»

Для спрощення адміністрування на сервері PostgreSQL в базовий комплект установки входить такий інструмент як pgAdmin. Він являє собою графічний клієнт для роботи з сервером, через який користувач в зручному вигляді може створювати, видаляти, змінювати бази даних і управляти ними.

2.3. Побудова серверної сторони на базі фреймворку Spring Boot

Spring Boot Framework — це програмний каркас (фреймворк) з відкритим кодом та контейнер з підтримкою інверсії управління для платформи Java.

Основні особливості Spring Boot Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав'язує якоїсь конкретної моделі програмування, Spring Framework став популярним в спільноті Java як альтернатива, або навіть доповнення моделі Enterprise JavaBean (EJB).

До складу Spring Boot входить багато підпроектів, «заточених» під певну функціональність (SpringMVC, Spring Security, SpringData і ін.).

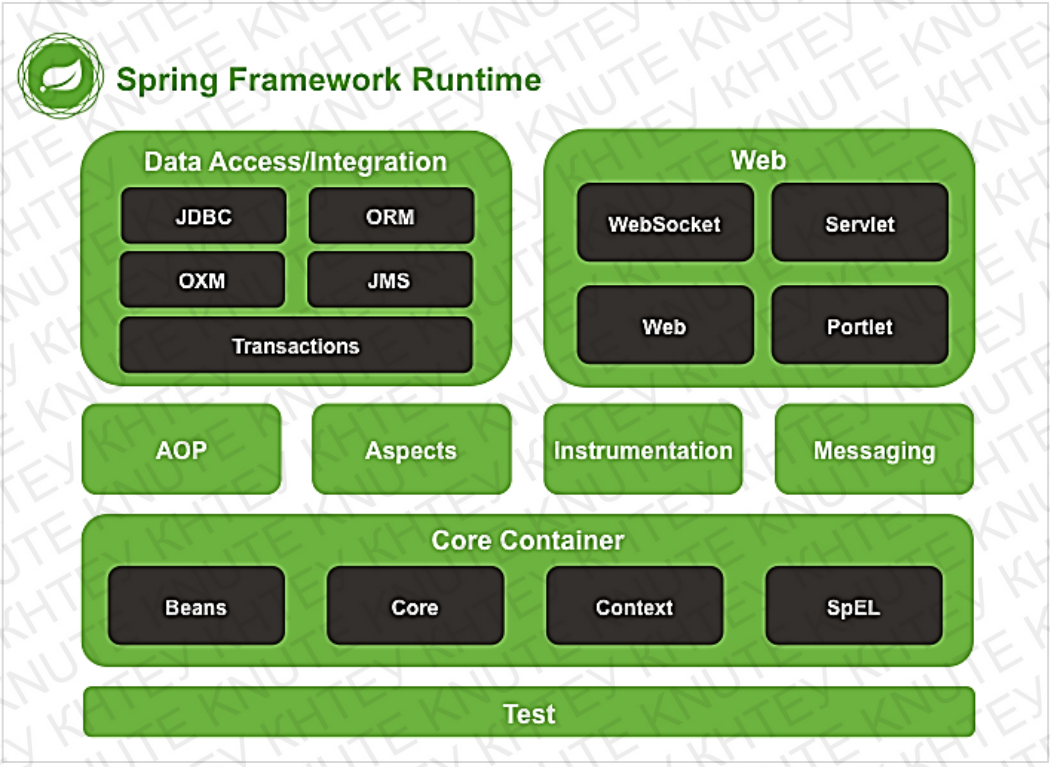

Рис. 2.3. Схематична структура фреймворку «Spring Boot»

В Spring Boot налаштування компонентів відділені від програмного коду. Винесення конфігурації (управління залежностями) в окремий файл полегшує наступні зміни в проекті (заміна реалізацій):

- поліпшена можливість тестування; коли класи проектуються на основі DI і інтерфейсів, стає можливою проста заміна залежностей (фейковий реалізаціями) при тестуванні;
- можливість програмування в декларативному стилі за допомогою анотацій зменшує кількість коду в додатку;
- відмінна інтеграція з технологіями доступу до даних.

2.4. Побудова веб-компонентів засобами бібліотеки ReactJS

ReactJS (або React) — відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці

односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. React обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC), і може бути використане у поєднанні з іншими JavaScript бібліотеками або в великих фреймворках MVC, таких як AngularJS. Він також може бути використаний з React на основі надбудов, щоб піклуватися про частини без користувацького інтерфейсу побудови веб-застосунків. Як бібліотеку інтерфейсу користувача React найчастіше використовують разом з іншими бібліотеками, такими як Redux.

В даний час React використовують Netflix, Yahoo, Airbnb, Sony, Atlassian, Приват24, Keeper, Salesforce та інші.

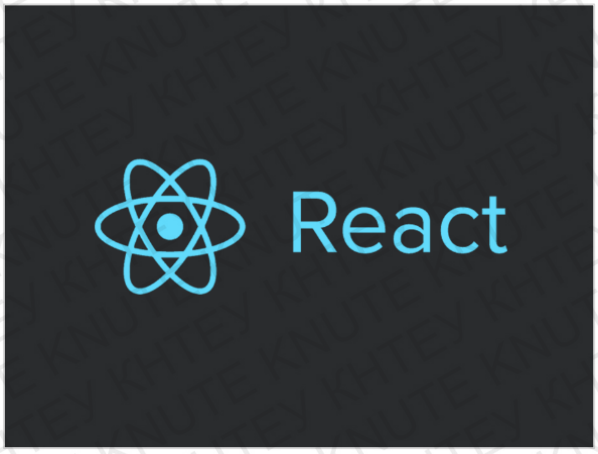

Рис. 2.4. Логотип JS бібліотеки «React»

Компоненти React зазвичай написані на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому JavaScript. JSX нагадує іншу мову, яку створили у компанії Фейсбук.

2.5. Контроль версій ПП

Життєвий цикл кожної успішної комп'ютерної програми може бути дуже довгим, і зміни в програмі бувають різними — від виправлення помилки до повного переписування.

Під час розробки програмного продукту виникає необхідність моніторити зміну коду, зберігати його та мати вічний надійний доступ. Для правильної роботи з кодом, а саме його збереження та контролю використовуються системи контролю версій.

Система контролю версій (СКВ) – це система, що записує зміни у файл або набір файлів протягом деякого часу з можливістю повернення до певної їх версії пізніше.

Однією з найпопулярніших та найзручніших для колективної розробки ПП є розподілена/децентралізована СКВ «Git» – це програмний інструмент, основне завдання якого – зберігання коду та історії його у репозиторії коду.

Рис.2.5. Логотип СКВ «Git»

Користувачі Git мають можливість користуватися онлайн сервісом для своїх публічних чи приватних репозиторіїв *GitHub*.

Рис.2.6. Логотип онлайн-сервісу «GitHub»

GitHub використовується як веб-сервіс для майбутнього хостингу проектів, а також як соціальна мережа для розробників. Користувачі можуть заливати свої роботи та публікувати їх на сайті. Кожен користувач створює свій окремий репозиторій та публікує свої роботи. GitHub на даний момент є найпопулярнішим сервісом такого виду. Для проектів з відкритим кодом (публічних) використання є безкоштовним. У випадку закритого середовища для роботи – платним.

2.6. Управління проектом

Проект – це обмежений часовими рамками процес, що має визначений початок та кінець, зазвичай обмежений датою, але також може обмежуватися фінансуванням або досягненням результатів, який здійснюється для реалізації унікальних цілей та завдань. Головним завданням проектного управління є досягнення всіх цілей та виконання завдань проекту.

Система управління проектами (СУП) – визначення для комплексного програмного забезпечення, що включає в себе програми для планування завдань, складання розпису, швидкого управління, документування та адміністрування системи.

Для управління випускним кваліфікаційним проектом було обрано таку систему управління проектами як «*VivifyScrum*».

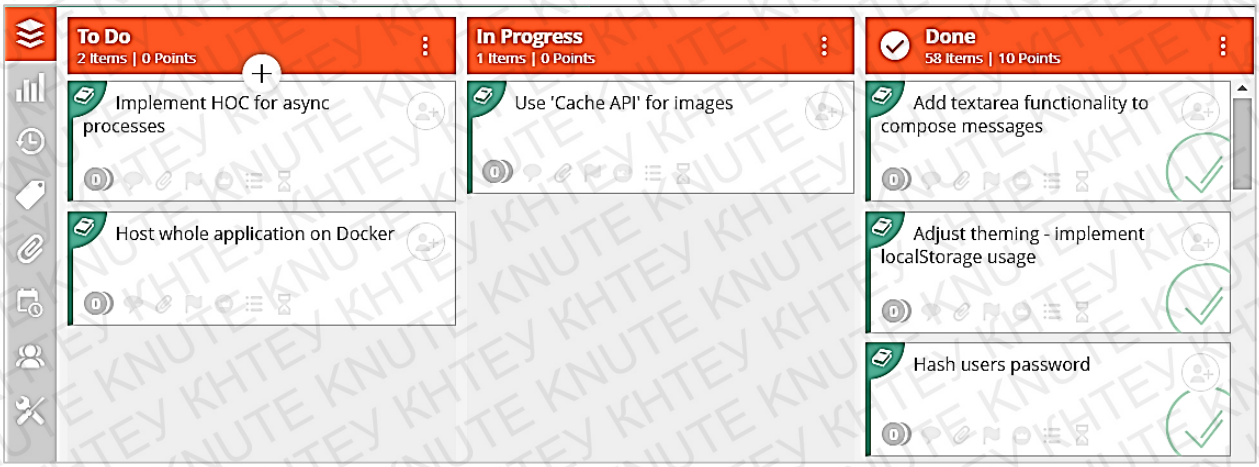

Рис. 2.7. Інтерфейс СУП «VivifyScrum»

Однією з переваг цієї СУП є те, що вона безкоштовна. Окрім цього «VivifyScrum» надає велику кількість необхідної інформації, такої як:

- ❖ інформацію про розподіл ресурсів;
- ❖ огляд інформації про терміни виконання завдань;
- ❖ інформація про робоче навантаження;
- ❖ детальна технічна інформація щодо завдань.

2.7. Хостинг ПП на хмарній платформі

Веб-хостинг – це послуга з розміщення сайту на сервері, щоб користувачі могли бачити його в інтернеті.

Хостинг-провайдери зберігають файли сайту на потужних комп'ютерах, що називаються веб-серверами. Коли користувачі набирають адресу сайту, Всесвітня мережа під'єднує їх до хостинг-серверу з усіма файлами сайту і надсилає інформацію на їхні комп'ютери, щоб вони могли переглядати сторінки сайту. Важливим є забезпечення швидкого та безперебійного інтернет-з'єднання. Безпечність, швидкість, стабільна робота, надійність – характеристики, без яких неможливо уявити хороший хостинг-сервер.

Heroku – хмарна PaaS-платформа, що підтримує ряд мов програмування. Компанією Heroku володіє Salesforce.com. Heroku, одна з перших хмарних платформ, з'явилась в червні 2007 року і спочатку підтримувала тільки мову програмування Ruby, але на даний момент список підтримуваних мов також включає в себе Java, Node.js, Scala, Clojure, Python і PHP. На серверах Heroku використовуються операційні системи Debian або Ubuntu.

Рис. 2.8. Логотип хмарної платформи «Heroku»

Програми, що працюють на Heroku, використовують також DNS-сервер Heroku (зазвичай додатки мають доменне ім'я виду «*ім'я_додатку.herokuapp.com*»). Для кожної програми виділяється кілька незалежних віртуальних процесів, які називаються «dynos». Вони розподілені по спеціальній віртуальній сітці («dynos grid»), яка складається з декількох серверів. Heroku також має систему контролю версій Git.

Висновки до Розділу 2

На стадії планування розробки ПП було сформовано наступні характеристики додатку та обрано інструменти для його реалізації:

Таблиця 2.1.

ТЕХНОЛОГІЇ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

Назва	***Деталі***
Система управління базами даних	СУБД «*PostgreSQL*»
Технологія для побудови серверної сторони додатку	Програмна мова *«Java»* та фреймворк *«Spring Boot»*
Технологія для побудови *frontend* сторони	JS бібліотека «*ReactJS*»
Хостинг-провайдер	PaaS платформа *«Heroku»*

Таблиця 2.2.

ІНСТРУМЕНТИ ДЛЯ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ПРОДУКТУ

Інструмент	***Деталі***
Інтегроване середовище розробки	ІСР «*IntelliJ IDEA – Ultimate Edition*»
Система контролю версій	СКВ *«Git»*
Управління проектом	СУП *«VivifyScrum»*

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1. Структура таблиць в БД

База даних ПП «xMessenger» складається з чотирьох взаємозв'язаних таблиць (*див. рис. 3.1.).*

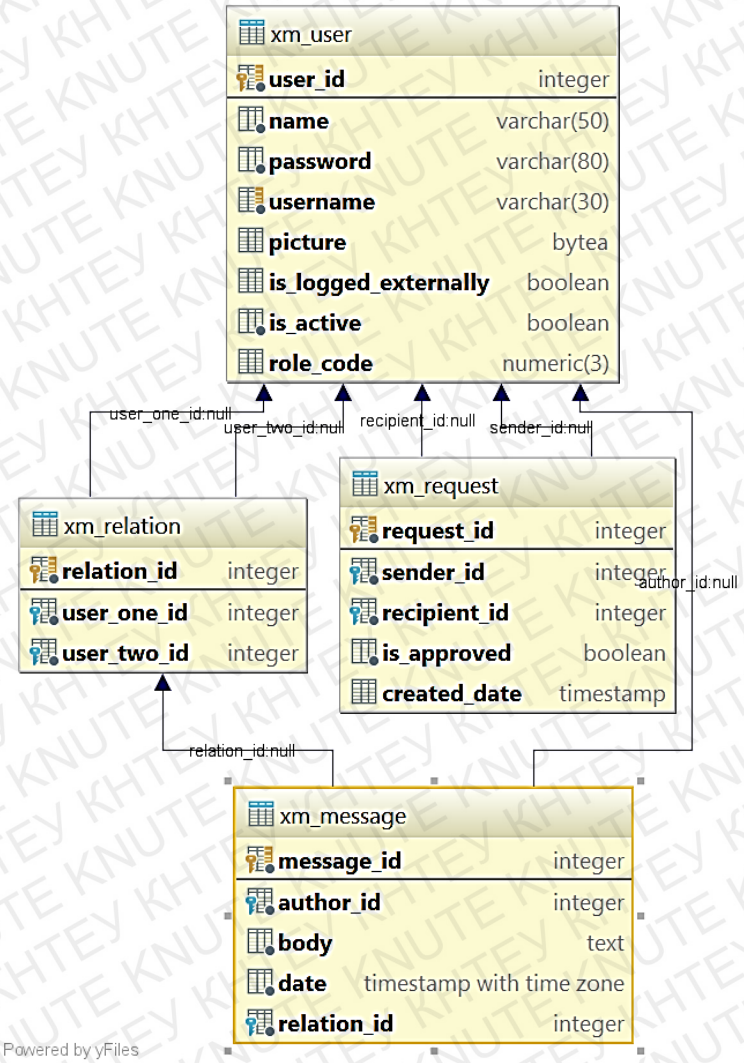

Рис. 3.1. Схема зв'язків таблиць в БД

Семантика кожної таблиці унікальна, оскільки містить різний тип інформації. Опис даних, що зберігаються в кожній таблиці наведено у таблиці 3.1.

Таблиця 3.1.

ОПИС ТАБЛИЦЬ В БД

Таблиця	**Опис**
xm_user	Детальна інформація про користувачів системи: ім’я, нікнейм, фото, пароль для входу, інша системні дані
xm_relation	Тут зберігається інформація щодо зв’язків між користувачами системи («*relation» з англ. перекладається як «відношення»*)
xm_request	Дані щодо існуючих запитів «дружби» адресовані одному користувачеві від іншого
xm_message	Місце збереження повідомлень (контент, дата відправлення, автор, унікальний ідентифікатор)

3.2. Приклади реалізації серверної сторони додатку

Першим кроком розробки *backend* було створення основних сутностей бази даних додатку, та проведено їх об’єктно-реляційне зв’язування. Як приклад розглянуто лістинг на рисунку 3.2.

```
6   @javax.persistence.Entity
7   @Table(name = "`relation`")
8   public class Relation {
9
10      @Id
11      @GeneratedValue(strategy = GenerationType.AUTO)
12      @Column(name = "rid")
13      private Integer id;
14
15      @ManyToOne
16      @JoinColumn(name = "userone")
17      private User userOne;
18
19      @ManyToOne
20      @JoinColumn(name = "usertwo")
21      private User userTwo;
22
```

Рис. 3.2. Лістинг класу Relation з використанням ORM-анотацій

Це вже описаний POJO-об’єкт, поля якого анотовані анотаціями об’єктно-реляційного відображення, що дозволяє зв’язати поля об’єктів з таблицями бази даних та їх атрибутами.

В даному прикладі добре показані можливості фреймворку. Основні з них – організація відношень типу багато-до-одного (анотація *@ManyToOne*), багато-до-багатьох (анотація *@ManyToMany*). Схожим способом імплементовані всі інші сутності додатку, що формують модель даних.

У шарі моделі даних, поверх шару POJO-об’єктів, є шар репозиторіїв. Шар репозиторіїв розташований між шаром домену та шаром даних. Концептуально, репозиторій інкапсулює набір об’єктів, збережених у сховищі і операції, виконані над цими об’єктами. Це надає більш об’єктно-орієнтований вигляд шару даних. Spring Data JPA надає ряд інтерфейсів, які можна наслідувати для спрощення реалізації цього патерну. Одним із основних інтерфейсів репозиторіїв є інтерфейс CrudRepository.

```
@NoRepositoryBean
public interface CrudRepository<T, ID extends Serializable> extends Repository<T, ID> {
    <S extends T> S save(S var1);

    <S extends T> Iterable<S> save(Iterable<S> var1);

    T findOne(ID var1);

    boolean exists(ID var1);

    Iterable<T> findAll();

    Iterable<T> findAll(Iterable<ID> var1);

    long count();

    void delete(ID var1);

    void delete(T var1);

    void delete(Iterable<? extends T> var1);

    void deleteAll();
}
```

Рис. 3.3. Інтерфейс CrudRepository

У реалізації додатку для кожної з сутностей створені репозиотрії. Це рішення полегшує подальшу розробку, адже при подальшому проектуванні шару контролерів та логіки можна мислити в звичному об’єктному контексті, не замислюючись над тонкощами реалізації шару даних (див. Додаток А).

HTTP запит, надісланий клієнтом створеному REST-сервісу перш за все обробляється *DispatcherServlet*-ом.

```
@RestController
@RequestMapping("/chats/{chatId}")
public class ChattingController {
    private final HttpSession session;
    private final ApplicationEventPublisher publisher;
    private final ChatterFlowExecutor flowExecutor;

    @Autowired
    public ChattingController(HttpSession session, ApplicationEventPublisher

    @RequestMapping(value = "/", method = RequestMethod.GET)
    public List<Chat> getChats() {
        User user = SessionManager.take(this.session).getCurrentUser();
        return this.flowExecutor.getChats(user);
    }

    @RequestMapping(value = "/messages", method = RequestMethod.GET)
    public List<Message> getMessagesForChat(@PathVariable Integer chatId) {
        Chat chat = new Chat(chatId);
        return this.flowExecutor.getMessages(chat);
    }
```

Рис. 3.4. Типовий REST контролер

Після обробки DispathcerServlet знаходить потрібний контролер запитуючи його спеціального класу, котрий містить список відповідності ресурсів та контролерів і викликає його відповідні методи, в залежності від типу запиту.

Контролери та DispatcherServlet-и контролюються Spring Data REST. Після того як контролеру делеговано запит, викликається функціонал, специфічний для контролера.

Spring Boot містить в цілу купу внутрішніх модулів, кожен з яких спрямований на вирішення тих, чи інших проблем пов'язаних з ентерпрайз додатками. Відповідно, для хешування паролів та написання юніт тестів були застосовані такі модулі як *Spring Security* and *Spring Testing*. Приклади реалізацій наведено у Додатках Б і В.

3.3. Детальний огляд та структура веб-компонентів

Компоненти – основні будівельні блоки React. По суті це набір з HTML, CSS та JS, який має деякі внутрішні особливості, які властиві тільки компонентам. Мені подобається порівнювати компоненти React з пиріжками

з начинкою, тільки зі світу веб-технологій. Вони можуть містити все, чого ви тільки забажаєте, й бути огорнуті в оболонку, що легко компонується. Компоненти реалізуються на чистому JavaScript, або ж використовують JSX.

Для прикладу розберемо структуру компонента *Avatar.js* – його основне завдання відображати фотокартку користувача та зумити її при натисканні на неї (*англ. «zoom in»*) .

Avatar.js

```
1   import ...
7
8   class Avatar extends React.Component {
9       constructor(props) {...}
14
15      componentDidMount() {...}
30
31      componentDidUpdate(prevProps) {...}
39
40      handleImageClick(event) {...}
49
50      handleImageRef(element) {...}
58
59      render() {
60          const {user, className, isScalable = false} = this.props,
61              primaryClassName = Utility.join("slds-avatar", isScalable && "scalable-image"),
62              lazyPic = UserService.getLazyLoadPicture(), imgUrl = UserService.getPictureUrl(user);
63          return (
64              <span className={Utility.join(primaryClassName, className)}>
65                  <img title={user.name} alt={user.name}
66                       src={lazyPic} data-src={imgUrl}
67                       ref={this.handleImageRef}
68                       onClick={this.handleImageClick}/>
69              </span>
70          );
71      }
72  }
```

Рис. 3.5. Сирцевий код компоненти Avatar.js

Нижче приведено теми важливі для фундаментального розуміння React:

- *JSX* – дозволяє нам використовувати HTML-подібний синтаксис, що буде трансформовано у легкі JavaScript об'єкти.
- *React.Component* – спосіб створення нового компонента.
- *render()* – визначення UI окремого компонента.
- *state* – внутрішнє сховище даних (об'єктів) компонента.
- *constructor()* – встановлює початковий стан компонента.
- *setState* – допоміжний метод, що використовується для оновлення стану компонента і повторного рендерингу інтерфейсу користувача.
- *props* – дані, що передаються від батьківського до дочірнього компонента.

Життєвий цикл компонента:

- ✓ *componentDidMount* – виконується після встановлення (mount) компонента.
- ✓ *componentWillUnmount* – виконується перед тим, як компонент буде знищено.

Для передачі даних (повідомлень) у режимі реального часу було використано технологію веб-сокет (*англ. WebSocket*). WebSocket – це протокол, що призначений для обміну інформацією між браузером та веб-сервером в режимі реального часу. Він забезпечує двонаправлений повнодуплексний канал зв'язку через один TCP-сокет. WebSocket спроектовано для втілення у веб-браузерах та веб-серверах, але може також використовуватись будь-яким клієнт-серверним застосунком. Прикладний програмний інтерфейс WebSocket був стандартизований W3C.

Зі сторони клієнта – веб-переглядача – було використано JS бібліотеку SockJs для реалізації двостороннього зв’язку (клієнт-сервер-клієнт).

websocket-listener.js

```
'use strict';

require('stompjs/lib/stomp.js');

const SockJS = require('sockjs-client'), register = registrations => {
    let socket = SockJS('/client'), stompClient = Stomp.over(socket);
    stompClient.debug = null; // disable logging;
    stompClient.connect({}, _ => {
        registrations.forEach(registration => {
            stompClient.subscribe(registration.route, registration.callback);
        });
    });
};

module.exports = {
    register: register
};
```

Рис. 3.6. Реєстратор веб-сокет слухачів (websocket listeners) на стороні клієнта (браузера)

Додаткові приклади веб-компонентів наведено у Додатках Г та Д.

3.4. Управління ПП на хмарній платформі Heroku

Heroku – це прекрасна хмарна платформа, доступна для всіх розробників та компаній як хостинг-сервіс на рівні підприємства, розроблений відповідно

до вимог хостингу. З веб-сайтів із захопленням, усе до високого трафіку, критично важливих бізнес-сайтів, Heroku може впоратись з усіма. Найкраще, їх структура ціноутворення включає в себе безкоштовний рівень, який більше, ніж здатний запустити невеликий веб-сайт, такий як портфоліо.

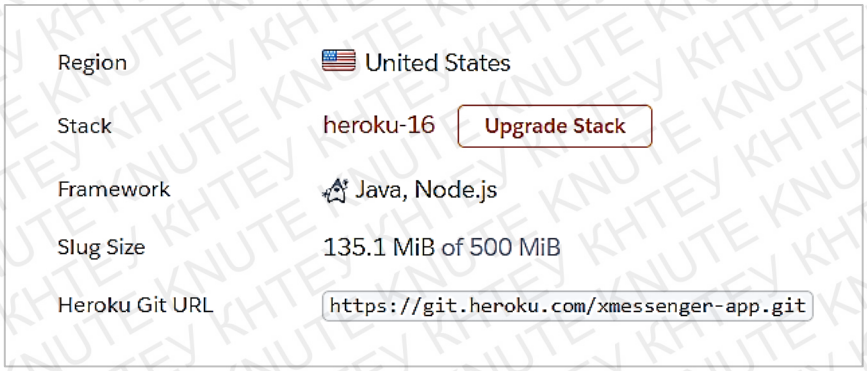

Рис. 3.7. Дані про додаток «xMessenger» зі сторони Heroku

Heroku використовує популярний інструмент керування вихідними кодами Git, як їх механізм управління розгортаннями на платформі. Все, що потрібно для початку – це проект, встановлений Git та обліковий запис Heroku, який можна отримати, відвідавши сторінку реєстрації.

Heroku безкоштовно надає один *dyno* контейнер, який здатний запустити єдиний екземпляр програми та помірний трафік до цього екземпляра.

Одним із ключових кроків до створення веб-додатку на цій платформі було обрання доменного імені. Всі безкоштовно розміщені додатки на Heroku створюються як суб-домени *.herokuapp.com*. В той час, як реєстрація унікальної адреси вимагає грошових інвестицій.

Додаток *xMessenger* було зареєстровано за наступною адресою: https://xmessenger-app.herokuapp.com (на момент реєстрації це був єдиний вільний варіант).

Висновки до розділу 3

У цьому розділі було розглянуто внутрішню структуру додатку «*xMessenger*». Структура БД ПП включає такі речі, як:

- дані про користувачів;

- інформація про повідомлення;
- зв’язки між користувачами системи.

Серверна сторона додатку написана на базі фреймворку «Spring Boot», мовою програмування якого є Java. Основна перевага «*Spring Boot*» – зручний підхід до впровадження залежностей (*англ. Dependency injection, D*I) – шаблон проектування програмного забезпечення, що передбачає надання зовнішньої залежності програмному компоненту, використовуючи «інверсію управління» (*англ. Inversion of control, IoC*) для розв'язання (отримання) залежностей.

Frontend сторона реалізована засобами відкритої JavaScript бібліотеки для створення інтерфейсів користувача *React*. Його мета полягає в тому, щоб бути швидким, простим, масштабованим. *React* обробляє тільки користувацький інтерфейс у застосунках. Це відповідає видові у шаблоні модель-вид-контролер (MVC).

Одним із найважливіших питань постало питання розміщення додатку у мережі Інтернет. В якості хостинг-провайдера було обрано PaaS-платформу Heroku. Перевагою такого вибору стала її доступність та простота налаштування.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Слід визнати, що у сучасному суспільстві одним із найбільш зручних та надійних засобів зв’язку є месенджери. Їх популярність на сьогоднішній день дійсно колосальна, тому розробники мають мільярдні прибутки.

Месенджери дають безкоштовну або платну можливість не тільки обмінюватися інформацією. Деякі програми надають шанс дзвонити на мобільні й стаціонарні телефони в будь-яку точку світу. Єдиний мінус такого спілкування – наявність підключення до мережі Інтернет. Якщо користувач, з яким ви хочете зв’язатися, знаходиться поза зоною дії мережі, то розмови не вийде.

Розробка браузерного месенджера – це доволі важкий та довгий процес, що вимагає скурпульозного підходу до всіх деталей кінцевого програмного продукту.

Метою даного проекту було опанування сучасних технологій для розробки веб-додатків та їх закріплення на практиці, а також набуття навичок роботи з хмарними платформами.

У функціонал кінцевого програмного продукту було закладено як основні/базові вимоги (обмін повідомленнями у режимі реального часу), так і додаткові (інтеграція з платформою Google).

Набір інструментів та технологій для реалізації також заслуговує окремої уваги. Особливого акценту слід надати бібліотеці для побудови веб-компонентів React та фреймворкові *«Spring Boot»*. Останній користується широким попитом серед *backend* технологій для малих та середній додатків/мікросервісів.

					КНТЕУ 6.050103 07-11.БР			
					Розробка браузерного месенджера на хмарних платформах	*Стадія*	*Аркуш*	*Аркушів*
Зм	*Аркуш*	*№ докум.*	*Підпис*	*Дата*		*ВП*	*24*	*27*
Зав. каф.		*Криворучко О.В.*						
Керівник		*Криворучко О.В.*						
Гарант		*Цензура М.О.*			*ВИСНОВКИ ТА ПРОПОЗИЦІЇ*	*Факультет обліку, аудиту та інформаційних систем, 4 курс, 7 група*		
Розроб.		*Мельнічук А.О.*						

Основою будь-якої успішної кампанії веб-додатку є його хостинг, або ж хмарна платформа, що обслуговує та керує його ресурсами. Heroku – це прекрасна хмарна платформа, доступна для всіх розробників та компаній як хостинг-сервіс на рівні підприємства, розроблений відповідно до вимог хостингу. З веб-сайтів із захопленням, усе до високого трафіку, критично важливих бізнес-сайтів, Heroku може впоратись з усіма.

Додаток *xMessenger* було зареєстровано за наступною адресою: https://xmessenger-app.herokuapp.com (на момент реєстрації це був єдиний вільний варіант).

Питання щодо безпеки та конфіденційності даних користувачів підіймалося неодноразово. Тут слід виділити декілька аспектів:

1. Хешування паролів – використовуючи модуль безпеки *Spring Boot* було застосовано алгоритми хешування паролів користувачів. Це означає, що навіть якщо ціла база даних буде скомпрометована, то у зловмисника будуть найменші шанси аби підібрати пароль конкретного користувача.
2. Наявність SSL-сертифіката – забезпечує конфіденційність обміну даними між клієнтом і сервером, що використовують TCP/IP, причому для шифрування використовується асиметричний алгоритм з відкритим ключем. SSL сертифікат безкоштовно наданий хостинг-провайдером – платформою Heroku.

У перспективі, великою оптимізацією було би розподілення монолітної системи на два мікросервіси: один для візуальної репрезентації для користувачів/адміністраторів (побудований на *Node* або *Express*) та інший для експозиції захищеного API для роботи з даними.

Наостанок хотілося б виділити ще одну істотну ознаку додатку – його адаптивний веб-дизайн. Метою адаптивного веб-дизайну є практичне відображення інформації та зручна навігація на всіх пристроях із доступом до інтернету (від стаціонарних ПК до мобільних телефонів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ren Chevance. Server Architectures: Multiprocessors, Clusters, Parallel Systems, Web Servers, Storage Solutions / Ren Chevance – Elsevier/Digital Press, 2005. – 690 p.
2. Erich Gamma. Design Patterns: Elements of Reusable Object-Oriented
3. Software / Erich Gamma, Richard Helm, Ralph Johnson et al. – AddisonWesley Professional, 1994. – 458 p.
4. Martin Fowler. Patterns of Enterprise Application Architecture / Martin
5. Fowler. – Addison-Wesley Professional, 2002. – 649 p.
6. Len Bass. Software Architecture in Practice / Len Bass, Paul Clements,
7. Rick Kazman. – Addison-Wesley Professional, 2012. – 334 p.
8. Гонсалвес Э. Изучаем Java EE 7 / Э. Гонсалвес. – Appress, 2016
9. Peter Herzum. Business Component Factory : A Comprehensive Overview of Component-Based Development for the Enterprise / Peter Herzum,
10. Oliver Sims. - Wiley, 1999 – 257 p.
11. Jose Sandoval. RESTful Java Web Services / Jose Sandovel.

Інтернет-ресурси

12. Офіційний сайт компанії Oracle - https://docs.oracle.com
13. Сайт репозиторію Maven - https://mvnrepository.com
14. Офіційний сайт розробників IDE IntelliJ IDEA - https://www.jetbrains.com
15. Вільна енциклопедія – https://uk.wikipedia.org
16. Офіційний сайт Spring Boot Framework – https://spring.io
17. Офіційний сайт JS бібліотеки React – https://reactjs.org
18. Офіційний сайт CI утиліти Travis – https://travis-ci.org

19. Офіційний сайт хмарної платформи Heroku – https://www.heroku.com
20. Офіційний сайт стилістичної CSS бібліотеки SLDS – https://www.lightningdesignsystem.com/
21. Офіційний сайт СУБД PostgresSQL – https://www.postgresql.org
22. Інтернет ресурс «Radka.ua» – http://radka.in.ua
23. Офіційний сайт СКВ Git – https://git-scm.com
24. Блог *Віталія Подоби* – http://www.vitaliypodoba.com
25. Онлайн-сервіс для публікування/зберігання сирцевого коду GitHub – https://github.com
26. Layered Application, Microsoft Developers Network – Режим доступу: https://msdn.microsoft.com/en-us/library/ff650258.aspx
27. A multi-tier architecture for building RESTful Web services – Режим доступу: http://www.ibm.com/developerworks/library/wa-aj-multitier/ -
28. Jim Webber, Savas Parastatidis,Ian Robbinson. – O`Reilly Media.
29. Understanding the Three-Tier Architecture. Oracle Documentation – Режим доступу: http://docs.oracle.com.