МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

КУРСОВАЯ РАБОТА

по дисциплине: Разработка серверных частей интернет-ресурсов
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: 09.03.04 Программная инженерия

Тема: Серверная часть веб-приложения «Социальная сеть»

Студент: Титов Феликс Александрович
Группа: ИКБО-10-19
Работа представлена к защите 30.11.2021 (дата)_____/Титов Ф.А. /
(подпись и ф.и.о. студента)

Руководитель: доцент Лобанов Александр Анатольевич

Работа допущен к защите_____(дата)_____/Лобанов А.А./
(подпись и ф.и.о. рук-ля)

Оценка по итогам защиты: _____
_____ / _____, доцент, Лобанов Александр Анатольевич /
_____ / _____ /
(подписи, дата, ф.и.о., должность, звание, уч. степень двух преподавателей, принявших защиту)

М. РТУ МИРЭА. 2021 г.

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения (ИиППО)

ЗАДАНИЕ
на выполнение курсовой работы

по дисциплине: Разработка серверных частей интернет-ресурсов
по профилю: Разработка программных продуктов и проектирование информационных систем
направления профессиональной подготовки: Программная инженерия (09.03.04)

Студент: Титов Феликс Александрович
Группа: ИКБО-10-19
Срок представления к защите: 30.11.2021
Руководитель: Лобанов Александр Анатольевич, доцент

Тема: Серверная часть веб-приложения «Социальная сеть»

Исходные данные: используемые технологии: HTML5, CSS3, Java Spring, Jetbrains Intellij IDEA, PostgreSQL СУБД, наличие: межстраничной навигации, внешнего вида страниц, соответствующего современным стандартам веб-разработки, использование паттерна проектирования MVC. Нормативный документ: инструкция по организации и проведению курсового проектирования СМКО МИРЭА 7.5.1/04.И.05-18.

Перечень вопросов, подлежащих разработке, и обязательного графического материала:
1. Провести анализ предметной области разрабатываемого веб-приложения. 2. Обосновать выбор технологий разработки веб-приложения. 3. Разработать архитектуру веб-приложения на основе выбранного паттерна проектирования. 4. Реализовать слой серверной логики веб-приложения с применением выбранной технологии. 5. Реализовать слой логики базы данных. 6. Разработать слой клиентского представления веб-приложения 7. Создать презентацию по выполненной курсовой работе.

Руководителем произведён инструктаж по технике безопасности, противопожарной технике и правилам внутреннего распорядка.

Зав. кафедрой ИиППО: _____/Р. Г. Болбаков/, «4» 09 2021 г.

Задание на КР выдал: _____/А.А Лобанов/, «21» 05 2021 г.

Задание на КР получил: _____/Ф.А. Титов/, «21.» 09 2021 г.

УДК 004.4

Титов Ф.А. Web-приложение на тему «Социальная сеть»/ Курсовая работа по дисциплине «Разработка серверных частей интернет-ресурсов» профиля «Разработка программных продуктов и проектирование информационных систем» направления профессиональной подготовки бакалавриата 09.03.04 «Программная инженерия» (5-ый семестр) / руководитель доцент Лобанов А.А. / кафедра ИППО Института ИТ МИРЭА – с.39, табл. 1, ист. 28.

Целью курсовой работы является разработка серверной части приложения «Социальная сеть».

В рамках работы проведен, анализ предметной области, согласно выбранной теме, выбор технологий разработки, разработка архитектуры на основе выбранного шаблона проектирования, разработка модели базы данных и внутренней логики приложения тестирование и развертывание созданного приложения.

Titov F.A. Web-application on the topic «Social Network»/ Course work on the discipline "Development of server parts of Internet resources" profile "Development of software products and design of information systems" direction of professional training for bachelor's degree 09.03.04 "Software engineering" (4th semester) / docent Lobanov A.A. / Department of IOPS of the Institute of IT MIREA - p. 39, tab. 1, ist. 28.

The purpose of the course work is to develop the server part of the "Social Network" application.

As part of the work, an analysis of the subject area was carried out, according to the chosen topic, the choice of development technologies, the development of architecture based on the selected design pattern, the development of a database model and internal application logic, testing and deployment of the created application.

М. МИРЭА. Ин-т ИТ. Каф. ИиППО. 2021 г. Титов Ф.А.

Аннотация

Курсовая работа содержит 39 страниц отчёта, 13 иллюстраций, 1 таблицу, 28 использованных источников информации.

Целью данной курсовой работы является подробное описание процесса создания и разработки серверного программного приложения. Работа выполнена с использованием языка программирования Java, и фреймворком Spring, в частности Spring Data JPA, Spring MVC, Spring Security. В качестве системы управления базами данных использовалась PostgreSQL. Программное приложение поддерживает сохранение сессий пользователя в базе данных класса NoSQL – Redis.

Результатом данной курсовой работы является созданное серверное программное приложение.

Содержание

Глоссарий

БД – база данных

СУБД — система управления базами данных

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

IDE (Integrated Development Environment) – интегрированная среда разработки приложений

Твит – короткое сообщение в социальной сети Twitter

Dependency Injection – процесс предоставления внешней зависимости программному компоненту

SQL - декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных

UI – пользовательский интерфейс

MVC – шаблон проектирования

HTML – язык гипертекстовой разметки

CSS – каскадные таблицы стилей

Эксплойт – компьютерная программа или фрагмент кода, совершающие атаки на вычислительную систему

Handler – обработчик поступивших сигналов

Сервлет – интерфейс Java, реализация которого расширяет функциональные возможности сервера

API – интерфейс прикладного программирования

Шаблонизатор - программное обеспечение, позволяющее использовать HTML-шаблоны для генерации конечных HTML-страниц

Web - система доступа к связанным между собой документам на различных компьютерах, подключённых к одной сети

Бэкап – предварительно созданная копия данных для восстановления в случае потери оригинальных данных

Хештег – ключевое слово или несколько слов для сообщения, используемый в социальных сетях

Репозиторий – место, где хранятся и поддерживаются какие-либо данные

Введение

С развитием интернета и появлением на свет первых площадок для общения людей, нам мир разделился на онлайн и офлайн составляющие. Данные площадки получили название социальные сети, где люди могут поддерживать связь, находясь на разных континентах, делиться мнениями, узнавать новости и многие другие вещи. Прогресс не стоит на месте, это касается и социальных сетей, которые и по сей день продолжают развиваться.

Цель курсовой работы: разработать серверную часть веб-приложения «Социальная сеть».

Объект исследования: процесс разработки веб-приложений.

Предмет: языки программирования и разметки, а также связанные с ними технологии для разработки веб-приложений.

Задачи для достижения поставленной цели:

1. Проанализировать предметную область рассматриваемой темы, понять ее назначение, и рассмотреть существующие аналоги.
2. Рассмотреть функциональные возможности языков программирования и соответствующих им фреймворков.
3. Спроектировать структуру и внешний вид веб-приложения, проанализировать его и создать межстраничную навигацию.
4. Реализовать бизнес-логику веб-приложения и протестировать реализованный функционал.

При написании данной курсовой работы использовались метод теоретического исследования (анализ) и практический метод (моделирование).

В ходе выполнения курсовой работы должно было быть проведено освоение и углубление профессиональных компетенций, предусмотренных федеральным образовательным стандартом, а именно: ПК-1, в том числе ПК-1.2, ПК-1.12, ПК-1.16.

Выполнение курсовой работы опирается на положения СМКО МИРЭА 7.5.1/03.П.69-18.

1. Анализ предметной области

1.1 Назначение интернет-ресурса и его основной функционал

Социальная сеть – онлайн-платформа, которую используют для общения, поиска друзей, возможность делить своими мыслями и фактами из жизни. В качестве социальной сети можно рассматривать любое онлайн-сообщество, члены которого участвуют в обсуждениях различных тем. Социальные сети образуются читателями тематических сообществ. Одной из главных особенностей социальных сетей является поиск единомышленников по различным интересам. Многие социальные сети позволяют публиковать различные публикации в формате блогов. Пользователи могут объединять группу сообщений по теме или по их типу с использованием хештегов – ключевых слов или фраз, начинающихся со знака #.

В функционале пользователя есть возможность редактировать данные своего профиля, а также помимо создания сообщений и прикрепления к ним фотографий, возможность изменять их в будущем. Пользователи могут подписывать на других пользователей, и ставить отметку «Нравится» их сообщениям. Для того, чтобы пользователи могли выполнять вышеприведенный функционал пользователи заполняют форму регистрации на сайте социальной сети.

1.2 Аналоги разрабатываемого интернет-ресурса

При проведении анализа у функционала у уже существующих известных аналогов были рассмотрены две крупнейшие и известные на весь мир социальные сети – Twitter [1] и Facebook [2].

Facebook и Twitter – два гиганта социальных сетей, позволяющие нам – пользователям общаться с другими людьми в режиме онлайн, делиться своими мыслями, взглядами, идеями, новостями, информацией. Обе платформы представляют разные концепции социальных сетей. Кардинальные различия указанных социальных сетей приведены в таблице 1.2.1:

Таблица 1.2.1 Сравнение социальных сетей

Категория сравнения	Facebook	Twitter
Главная особенность	Социальная сеть с ведением профиля и общением с другими пользователями	Социальная сеть с поддержкой ведения микроблогов
Мгновенный обмен сообщениями	Да	Нет
Аудитория	Друзья	Подписчики
Срок службы контента	Статусы сохраняются навсегда	Из-за частых обновлений, взаимодействие является краткосрочным
Выражение взглядов на сообщения	Отметки «Нравится», комментарии	Отметки «Нравится», ретвиты

В Facebook можно добавить других пользователей в свой профиль, просто отправив им запрос на добавление в друзья, после чего вы сможете получить доступ к тому, что происходит в их жизни. При помощи данной платформы можно связываться миллионами людей. Люди используют эту сеть для многочисленных действий, таких как загрузка фотографий, обмен ссылками и видео, публикация статуса, обмен информацией, общение в чате, звонки и многое другое.

В Twitter люди делятся информацией посредством коротких сообщений, имеющих длину в 140 символов, которые называются твитами. Твит – это текстовый, фото и видео контент, который описывают мнение человека, идеи, новости от зарегистрированных пользователей. Это напоминает микроблоки, где каждый может выразить свою идею, и показать свою реакцию на конкретное сообщение в социальной сети.

Выводы к первому разделу

На основании рассмотренной выше информации можно сделать вывод, что в основном функционале приложения должна быть предусмотрена возможность создания и изменения сообщений для пользователя, возможность зарегистрировать и хранить сессии пользователей, просматривать сообщения интересующих пользователей, с возможностью подписаться на данных пользователей и реагировать на сообщения при помощи проставления отметок «Нравится» у сообщений.

Учитывая особенности данного веб-приложения, оно достаточно востребовано среди пользователей в наше время, поэтому интернет-ресурс будет пользоваться спросом при добавлении вышеуказанного функционала и следующим причинам:

1. Возможность быстро находить интересующую информацию благодаря хештегам;
2. Высказывание абсолютного любого мнения касаемо какой-либо тематики;
3. Нужное предоставление информации;
4. Возможность читать только интересующую информацию.

2. Выбор и обоснование технологий

2.1 Язык программирования и фреймворки

Для решения поставленной задачи, одной из важных составляющих является выбор языков программирования и необходимых для него фреймворков. Для создания различных web-приложений, существует множество языков подходящих для этой цели, таких как: PHP вместе с фреймворками Laravel или CodeIgniter, Java с популярными фреймворками Spring или PrimeFaces, Python с фреймворками Django или Flask и многих других.

Среди представленных выше языков программирования, мною был сделан выбор использовать язык Java вместе с фреймворком Spring , а именно Spring Boot. В отличие от других платформ, Spring акцентирует внимание на области функционирования приложений и дает широкий спектр дополнительных функций. Одной из его основных особенностей является использование шаблона Dependency Injection (внедрение зависимостей). Это помогает намного проще реализовать необходимую функциональность, а также позволяет создавать слабосвязанные классы, что делает их гораздо более универсальными. Spring Framework стремится придать гибкости приложение, а Spring Boot пытается сократить длину кода, и упростить разработку web-приложения при помощи аннотаций, что сокращает время на разработку. Spring Boot использует Spring Framework в качестве основы. Он упрощает зависимости Spring и запускает приложение прямо из командной строки, а также контролировать компоненты приложения.

2.2 Среда разработки

Для создания необходимого web-приложения необходима среда разработки, в которой можно следить за правильностью написания кода, разделения файлов с кодом в файловой структуре проекта, а также отладкой написанного кода. Среди сред разработки существует множество как коммерческих, так и бесплатных вариантов. Наиболее распространенными из них являются: IDE Intelij Idea от компании JetBrains [3], IDE NetBeans от

Apache Software Foundation и IDE Eclipse. Каждая из них обладает необходимым функционалом для разработки приложений на языке Java. Коммерческой средой разработки из перечисленных является IDE Intelij Idea, остальные являются бесплатными.

Среда разработки Eclipse имеет большое количество разработчиков, о чем говорит внушительное количество плагинов, но есть и обратная сторона – многие из них достаточно сомнительного качества и содержания.

Среда разработки NetBeans позиционируется производителем, как среда разработки, поддерживающая все новшества языка Java, и позволяет писать код без ошибок с помощью инструмента FindBug.

Среда разработки Intelij Idea одна из самых функциональных сред разработки на языке Java [4], также помогающая в разработке на фреймворке Spring [5], оснащенная системой интеллектуальной помощи в написании кода. Исходя из контекста, IDEA настраивает работу авто-дополнения и доступность инструментов и плагинов. Инструменты помогают разработчику ускорить разработку, указывая на повторяющиеся куски кода в программе, с помощью статических анализаторов, а также благодаря отладчику динамически прослеживать работу приложения и находить ошибки.

Мною было принято решение использовать Intelij Idea для разработки приложения согласно заданию курсовой работы, поскольку в ней присутствует множество удобных для разработчика инструментов, а минус коммерческого использования нивелирует возможность ее бесплатного использования для студентов.

2.3 Системы управления базами данных

Для разработки необходимого приложения, данными о пользователях социальной сети, о сообщениях, подписках и отметок «Нравится» необходимо управлять и хранить. С помощью системы управления базами данных можно управлять нужными для нас данными и хранить их, желательно в оптимизированном виде. Выбор СУБД является важным моментом при

создании собственного ресурса. Наиболее известными являются MySQL, PostgreSQL и Oracle.

MySQL является наиболее распространенной СУБД относящихся к реляционному типу. Данный тип представляет собой таблицы, в которых каждый столбец упорядочен и имеет определенное значение. Последовательность строк определяется последовательностью ввода информации в любом порядке. Таблицы с данными связаны между собой специальными отношениями, благодаря чему с данными из разных таблиц можно взаимодействовать. Как правило, эту систему управления базами данных определяют как хорошую, быструю и гибкую, рекомендованную к применению в небольших или средних проектах.

PostgreSQL - Эта свободно распространяемая система управления базами данных относится к объектно-реляционному типу СУБД [6]. В отличии от реляционной, данный тип поддерживают объектно-ориентированные языки программирования, благодаря чему есть возможность использовать основные принципы ООП: инкапсуляцию, полиморфизм и наследование и разработчику не придется переводить реляционную модель в объектную модель и наоборот. Как и в случае с MySQL, работа с PostgreSQL основывается на языке SQL, однако, в отличие от MySQL, PostgreSQL поддерживает стандарт SQL-2011. Эта СУБД не имеет ограничений ни по максимальному размеру базы данных, ни по максимуму записей или индексов в таблице.

Oracle – это система, отличающаяся стабильностью уже не один десяток лет, поэтому ее выбирают корпорации, для которых важна надежность восстановления после сбоев, отлаженная процедура бэкапа, возможность масштабирования и другие ценные возможности. К тому же эта СУБД обеспечивает отличную безопасность и эффектную защиту данных.

В отличие от других СУБД, стоимость покупки и использования Oracle достаточно высока, и именно это зачастую является значимым препятствием к ее использованию в небольших фирмах.

Из представленных СУБД для хранения у управления данными мною была выбрана PostgreSQL, поскольку обладает наиболее подходящими характеристиками для разработки на фоне конкурентов.

После авторизации пользователей в приложении, также необходимо хранить некоторую информацию о времени сеанса пользователей называемые сессиями. Они содержат информацию о запросах клиента на сервер. В момент авторизации создается соединение между клиентом и сервером, и у каждого клиента оно уникально. По умолчанию сессии хранятся в файлах, что не очень удобно, поскольку доступ к файлам происходит медленнее, чем к оперативной памяти, поэтому для их ранений была выбрана Redis – СУБД класса NoSQL, работающая со структурами данных типа ключ-значение [7]. Redis в первую очередь ориентирован на быстрое выполнение атомарных операций. Наиболее активное применение он находит в кэшировании и в реализации брокеров сообщений и очередей. Так же Redis периодически сохраняет данные на диск, что предотвращает потерю данных. Redis позволит удобным образом хранить данные о сессиях пользователя, а web-сервер будет знать об этих данных что позволит корректно авторизовать пользователя.

2.4 Паттерн проектирования

В Spring Framework используются множество паттернов проектирования, самым популярным из которых является MVC. При разработке систем с пользовательским интерфейсом, следуя паттерну MVC нужно разделять систему на три составные части. Их, в свою очередь, можно называть модулями или компонентами. У каждой составной компоненты будет свое предназначение.

- Model. Первая компонента/модуль — так называемая модель. Она содержит всю бизнес-логику приложения.
- View. Вторая часть системы — вид. Данный модуль отвечает за отображение данных пользователю. Все, что видит пользователь, генерируется видом.

- Controller. Третьим звеном данной цепи является контроллер. В нем хранится код, который отвечает за обработку действий пользователя (любое действие пользователя в системе обрабатывается в контроллере).

Модель — самая независимая часть системы. Настолько независимая, что она не должна ничего знать о модулях вид и контроллер. Модель настолько независима, что ее разработчики могут практически ничего не знать о виде и контроллере.

Основное предназначение вида — предоставлять информацию из модели в удобном для восприятия пользователя формате. Основное ограничение вида — он никак не должен изменять модель [8]. Основное предназначение контроллера — обрабатывать действия пользователя. Именно через контроллер пользователь вносит изменения в данные, которые хранятся в модели. Схему шаблона MVC можно увидеть на рисунке 2.4.1:

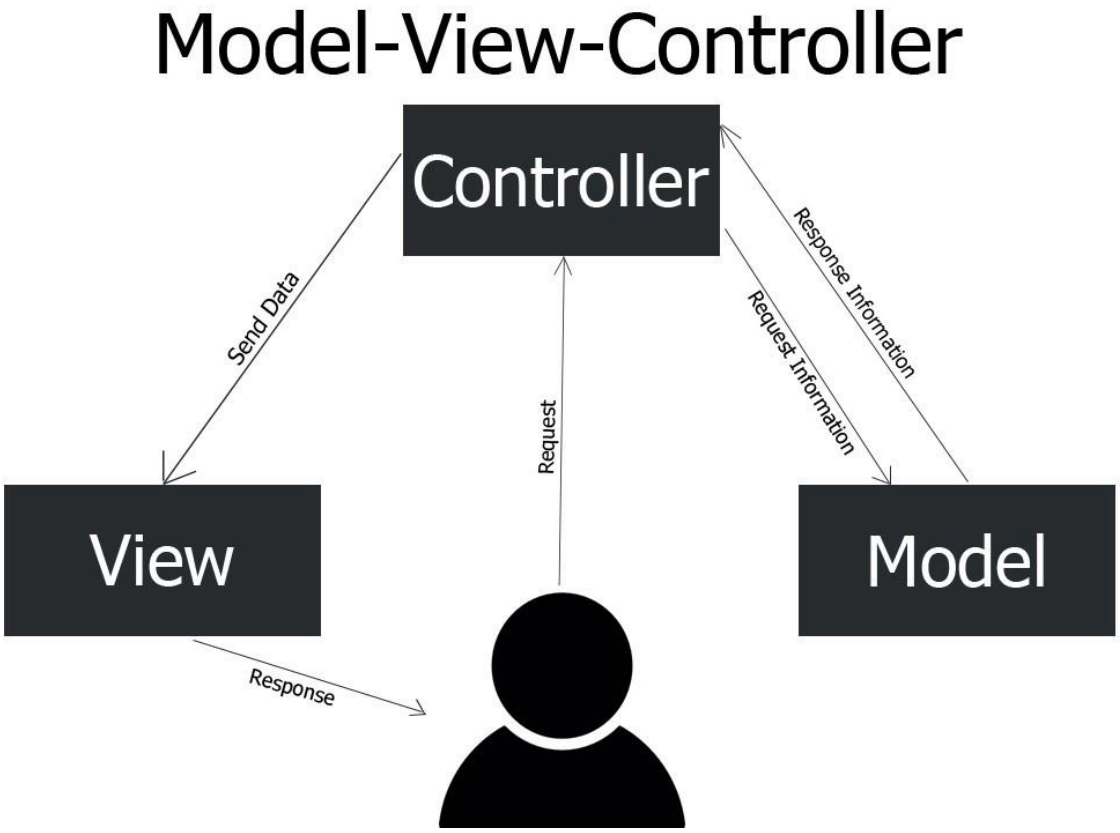

Рисунок 2.4.1 – Схема MVC

Помимо MVC, для поставленной задачи мною также был рассмотрен набор архитектурных принципов, называемых чистая архитектура (Clean Architecture). Она описывает общую архитектуру приложения, как различные уровни приложения: бизнес-объекты, варианты использования, докладчики, хранилище данных и пользовательский интерфейс взаимодействуют друг с другом. Эта архитектура, особенно популярная среди разработчиков Java, предназначена для упрощения создания стабильных приложений, даже когда внешние элементы, такие как пользовательский интерфейс, базы данных или внешние API, постоянно меняются.

Чистая архитектура обладает следующими преимуществами:

- Независимость от фреймворка: чистая архитектура не полагается на инструменты из какой-либо конкретной платформы и не использует фреймворк как зависимость где-либо в коде.
- Высокая степень тестируемости: чистая архитектура создается с нуля для тестирования.
- Независимость от базы данных: большая часть вашего приложения не будет знать или не должна знать, из какой базы данных оно черпает.
- Независимость от пользовательского интерфейса: каркасы пользовательского интерфейса существуют на самом внешнем уровне и поэтому являются просто представителем данных, передаваемых из внутренних слоев.

В самом просто виде, чистая архитектура представлена схемой на рисунке 2.4.2:

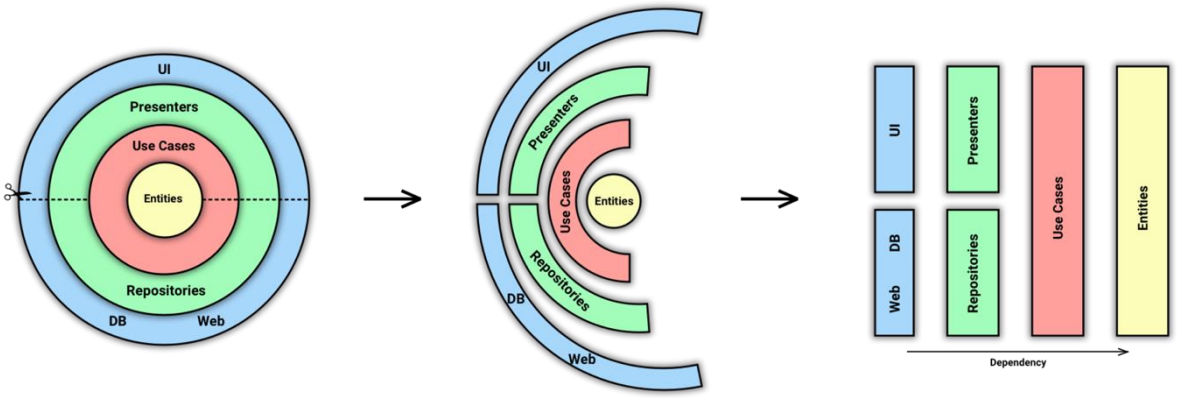

Рисунок 2.4.2 – Схема Clean Architecture

Этот вид лучше подходит для понимания инверсии зависимостей системы и для просмотра уровня абстракции для каждого уровня. Главный вывод из этой диаграммы состоит в том, что чистая архитектура позволяет внешним слоям зависеть от внутренних слоев, но не наоборот.

Цель данной концепции заключается в том, чтобы упростить разработку, развёртывание и поддержку программной системы. Главная стратегия такого упрощения в том, чтобы иметь возможность заменить какой-либо из инструментов, используемый при разработке [9].

В данной курсовой работе, наиболее удобным шаблоном проектирования, будет MVC. Набор архитектурных идей и принципов, которые представляет MVC, позволят упростить сопровождение кода: модификация одного из компонентов будет иметь минимальное на остальные, или не окажет его вовсе.

2.5 Отображение данных на стороне клиента

Среди выбранных технологий для реализации поставленной задачи, необходимо также отображать данные на стороне клиента, и дать возможность клиенту перемещаться по разделам сайта, для отображения разных данных, в зависимости от потребностей. Для реализации данного функционала была выбрана стандартная связка HTML5 и CSS3 вместе с фреймворком Bootstrap [10], для отображения страниц сайта, соответствующего современным стандартам разработки.

Для вывода данных необходим шаблонизатор, и наиболее удобным вариантом является Apache FreeMarker [11] — это механизм шаблонов: библиотека Java для создания текстового вывода. FreeMarker отображает подготовленные данные с помощью шаблонов. В шаблоне идёт уклон на том, как представить данные, а за пределами шаблона выбор, какие данные представлять. Схема шаблонизатора представлена на рисунке 2.5.1:

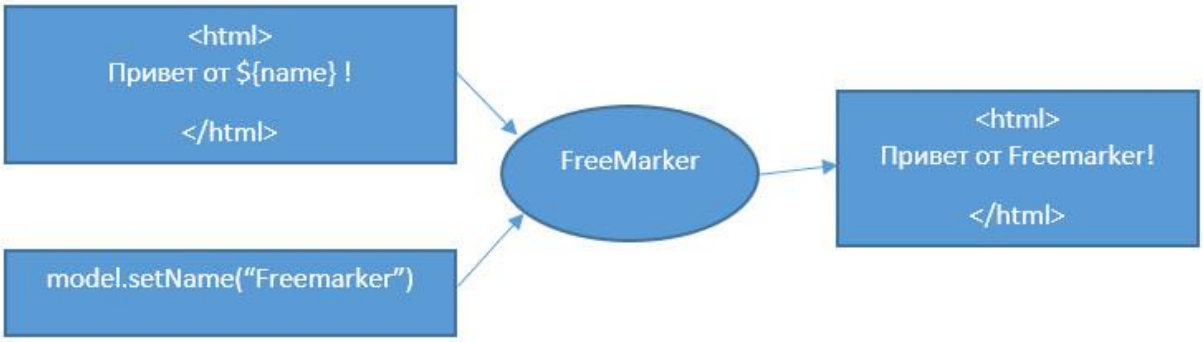

Рисунок 2.5.1 – Схема шаблонизатора FreeMarker

Выводы ко второму разделу

В результате рассмотренной выше информации, можно сделать вывод о выборе технологий, среди которых был выбран следующий стек:

- Язык программирования Java и фреймворк Spring
- СУБД PostgreSQL и Redis
- Паттерн проектирования MVC
- Среда разработки JetBrains Intelij IDEA
- HTML5, CSS3, Apache FreeMarker и Bootstrap

Данные технологии предоставят необходимый функционал для создания аналога рассмотренных социальных сетей.

3. Разработка архитектуры приложения на основе выбранного паттерна

3.1 Детальное описание используемого паттерна

В предыдущей главе для разработки приложения был выбран паттерн MVC, основная концепция которого, заключается в разделении функционала на 3 компонента:

- Модель. Этот уровень считается самым низким уровнем по сравнению с видом и контроллером. Он в первую очередь представляет данные пользователю и определяет хранение всех объектов данных приложения
- Представление. Этот уровень в основном связан с пользовательским интерфейсом и используется для обеспечения визуального представления модели MVC. Проще говоря, этот уровень касается отображения фактического вывода пользователю. Он также обрабатывает связь между пользователем (входами, запросами и т. д.) и контроллером.
- Контроллер. Этот уровень заботится об обработчике запросов**.** Его часто считают мозгом системы MVC - связью, так сказать, между пользователем и системой. Контроллер завершает цикл взятия пользовательского вывода, преобразования его в нужные сообщения и передачи их представлениям (UI).

Поскольку существует разделение кода между тремя уровнями, становится чрезвычайно легко разделить и организовать логику веб-приложений на крупномасштабные приложения (которыми необходимо управлять большие команды разработчиков) [12]. Основное преимущество использования таких методов кода заключается в том, что оно помогает быстро находить определенные части кода и позволяет легко добавлять новые функциональные возможности. Использование методологии MVC позволяет легко изменять все приложение. Добавление/обновление нового

типа видов упрощается в шаблоне MVC (поскольку один раздел не зависит от других разделов). Таким образом, любые изменения в определенном разделе приложения никогда не повлияют на всю архитектуру. Это, в свою очередь, поможет повысить гибкость и масштабируемость приложения.

При разделении кода между тремя уровнями, разработка веб-приложений с использованием модели MVC позволяет одному разработчику работать над определенным разделом (скажем, представлением), в то время как другой может работать над любым другим разделом (скажем, контроллером) одновременно. Это позволяет легко внедрить бизнес-логику, а также в четыре раза ускорить процесс разработки. Было отмечено, что по сравнению с другими моделями разработки модель MVC в конечном итоге показывает более высокую скорость разработки (до трех раз). В архитектуре MVC легко достижимо разработать различные компоненты представления для компонента модели. Это позволяет вам разрабатывать различные компоненты представления, тем самым ограничивая дублирование кода, поскольку оно разделяет данные и бизнес-логику.

MVC значительно упрощает процесс тестирования. Это облегчает отладку крупномасштабных приложений, так как в приложении структурно определены и правильно написаны несколько уровней. Таким образом, безотказно разрабатывает приложение с модульными тестами [13].

3.2 Архитектура web-приложения

Эффективность функционирования информационной системы во многом зависит от ее архитектуры. В настоящее время, перспективной является трехуровневая архитектура – одна из разновидностей архитектуры клиент-сервер. По сравнению с клиент-серверной архитектурой она имеет большую масштабируемость и конфигурируемость. Реализация приложений, доступных из веб-браузера, как правило, подразумевает развёртывание программного комплекса в трёхуровневой архитектуре. Трёхуровневая архитектура организует приложения в три логических и физических вычислительных уровня: уровень представления (пользовательский

интерфейс), уровень приложений (обработка данных происходит здесь) и уровень данных (хранение и управление данными). Поскольку каждый уровень работает в своей собственной инфраструктуре, он может разрабатываться одновременно отдельной командой разработчиков и может обновляться или масштабироваться по мере необходимости, не влияя на другие уровни, не говоря уже о реплицировании.

В трехуровневом приложении вся связь проходит через уровень приложения. Уровень представления и уровень данных не могут напрямую взаимодействовать друг с другом. Схему трехуровневой архитектуры можно увидеть на рисунке 3.1.1:

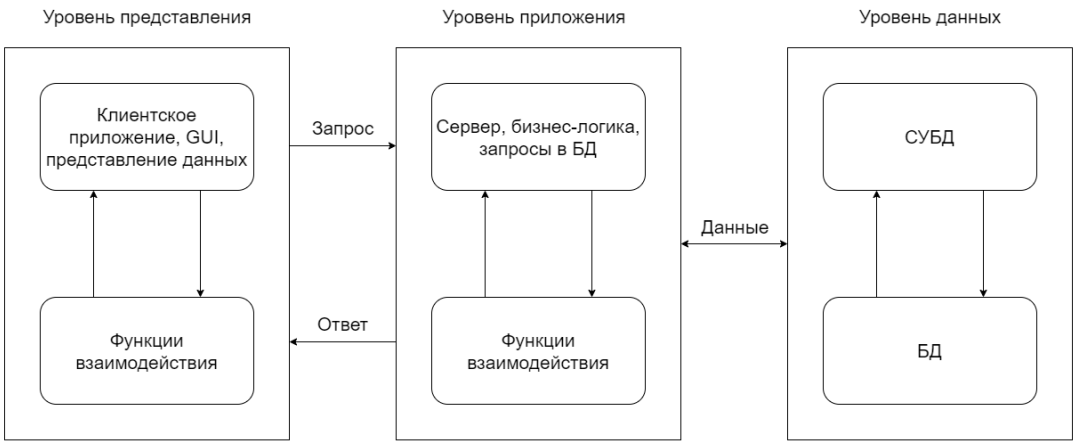

Рисунок 3.1.1 – Схема трехуровневой архитектуры

Уровень представления — это пользовательский интерфейс и слой связи приложения, на котором конечный пользователь взаимодействует с приложением. Его основная цель – отображать информацию для пользователя и собирать информацию от пользователя. Уровень презентации может работать в веб-браузере, в качестве настольного приложения или графического пользовательского интерфейса (GUI).

Уровень приложений, также известный как уровень логики или средний уровень, является сердцем приложения. На этом уровне информация, собранная на уровне представления, обрабатывается с использованием бизнес-логики. Уровень приложений также может добавлять, удалять или изменять данные на уровне данных.

Уровень данных, иногда называемый уровнем базы данных или уровнем доступа к данным, – это место, где хранится и управляется информация, обрабатываемая приложением. Его роль, как правило, выполняет СУБД. Она обеспечивает управление по созданию, использованию, управлению и контролю самих баз данных.

Как уже отмечалось ранее, главное преимущество трехуровневой архитектуры заключается в ее логическом и физическом разделении функций. Каждый уровень может работать на отдельной операционной системе и серверной платформе. Соответственно, не важно, где будут расположены платформы: в облаке, в одном здании, на разных континентах. В любом случае части распределенной системы должны быть соединены надежными и защищенными линиями связи. Что касается скорости передачи данных, то она в значительной степени зависит от важности соединения между двумя частями системы с точки зрения обработки и передачи данных и в меньшей степени от их удаленности. Службы каждого уровня могут быть настроены и оптимизированы без влияния на другие уровни.

Другие преимущества трехуровневой архитектуры:

- более быстрая разработка: отдельный уровень может разрабатываться одновременно разными командами, организация может быстрее выводить приложение на рынок, а программисты могут использовать новейшие и лучшие языки и инструменты в соответствии с каждым уровнем;
- улучшенная масштабируемость: отдельный уровень можно масштабировать независимо от других по мере необходимости;
- повышенная надежность: сбой на одном уровне с меньшей вероятностью повлияет на доступность или производительность других уровней;
- улучшенная безопасность: так как уровень представления и уровень данных не могут взаимодействовать напрямую, хорошо спроектированный уровень приложений может функционировать как своего

рода внутренний брандмауэр, предотвращающий инъекции SQL и другие вредоносные эксплойты [14].

Выводы к третьему разделу

На основании рассмотренной выше информации, можно сделать вывод, что выбранный шаблон проектирования MVC, со всеми его преимуществами, в полной мере позволяет реализовать серверную часть веб-приложения «Социальная сеть». Представленная архитектура, позволяет реализовать приложение, отвечающее требованиям, поставленных в задаче курсовой работы, а также в анализе предметной области.

4. Разработка серверной части интернет-ресурса

4.1 Реализация слоя логики базы данных

Проанализировав предметную область рассматриваемой темы, выбрав технологии для разработки интернет-ресурса, можно приступить к моделированию модели базы данных.

Основываясь на полученных знаний о том, как утроен основной функционал социальной сети, в базе данных обязательна таблица «users» (Пользователи). Сам пользователь должен обладать атрибутами, такими как: имя, пароль, email (электронная почта), активность пользователя на платформе. Это основные поля, которые принадлежат данной сущности. Также в социальных сетях есть возможность проверки адреса электронной почты при помощи кода активации, таким образом подтверждая факт того, что почта точно принадлежит данному пользователю. Для этого, у сущности пользователя будет также присутствовать поле с кодом активации. Таким образом, сущность «users» имеет следующие поля:

- id (обязательный атрибут, увеличивающийся на единицу)
- username (обязательное поле)
- password (обязательное поле)
- email (обязательное поле)
- active (обязательное поле)
- activation_code (необязательное поле)

Пользователи также могут содержать в себе некую роль. Это может быть обычный пользователь, или суперпользователь, обладающий большими правами в социальной сети, по сравнению с обычным пользователем. Для того, чтобы определять какой пользователь, обладает определенной ролью, удобно будет использовать отдельную таблицу, в которой будут храниться записи о ролях пользователя. Данная таблица будет называться user_roles и будет связана с таблицей пользователей при помощи связи «Один ко многим», поскольку один пользователь может иметь несколько ролей. Таблица будет

связана идентифицирующей связью при помощи внешнего ключа – уникального идентификатора пользователя (id).

Следуя концепции социальной сети Twitter, где пользователи могут оставлять сообщения, помечать их тэгами и добавлять фотографии, в базу данных следуют также добавить отдельную таблицу «message» (сообщения), которая будет обладать рядом собственных атрибутов:

- id (обязательный атрибут, увеличивающийся на единицу)
- text (обязательный атрибут)
- tag (необязательный атрибут)
- filename (необязательный атрибут)

Данная таблица будет также связана с таблицей пользователей при помощи связи «Один ко многим», поскольку один пользователь может иметь несколько сообщений. Связь между таблицами будет при помощи внешнего ключа – уникального идентификатора пользователя.

Во всех популярных социальных сетях предусмотрена возможность проставления отметок «Нравится» какому-либо сообщению, какого-либо пользователя. Данный механизм можно реализовать, добавив связь «Многие ко многим», между таблицей пользователей и таблицей сообщений. Таблица, которая будет являться посредником между ними, называемая message_likes, будет хранить в себе уникальный идентификатор пользователя, и уникальный идентификатор сообщения, таким образом находя информацию о том, кто поставил отметку «Нравится» и на какое именно сообщение отметка была поставлена.

Немало важным атрибутом социальных сетей, является механизм подписок одного пользователя, на другого интересного для него пользователя, таким образом быстрее узнавать информацию о сообщениях пользователя, на которого была подписка. Данные механизм также реализуется при помощи связи «Многие ко многим» только у самой таблицы пользователей. Таблица, дополнительно образующаяся данной связью, называемая user_subscribers,

будет хранить в себе два уникальных идентификатора пользователя, а именно того, кто подписывается, и на кого совершается подписка.

Итоговая физическая модель базы данных, показывающая типа данных каждого из полей таблицы, представлена на рисунке 4.1.1:

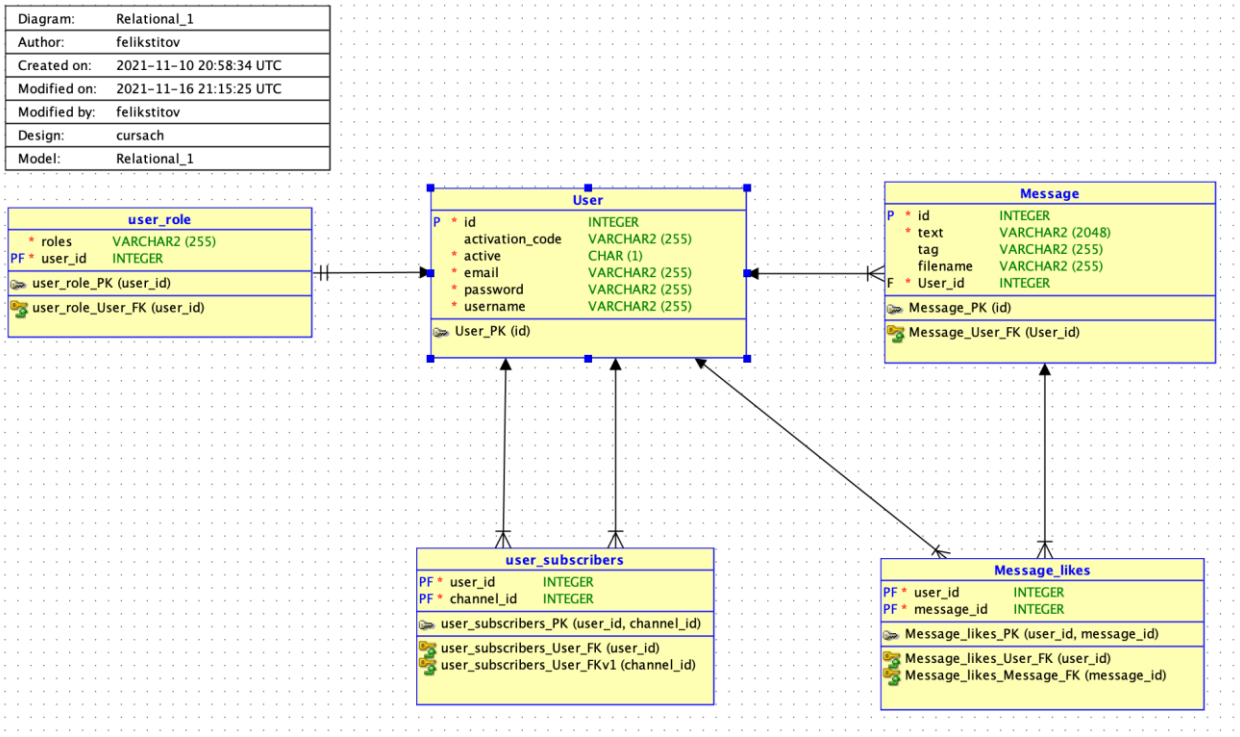

Рисунок 4.1.1 – Физическая модель базы данных

В базе данных, по умолчанию, содержится суперпользователь, с расширенными правами доступа, который также может давать данные права другим пользователям. Реализовано это при помощи SQL-скрипта, который запускается перед первичным запуском приложения, когда в новой среде создается база данных. Данный скрипт представлен на рисунке 4.1.2:

```
insert into users(id, active, password, username)
values (1,true,'admin','admin');

insert into user_role(user_id, roles)
values (1, 'USER'), (1, 'ADMIN');
```

Рисунок 4.1.2 – Создание суперпользователя приложения

4.2 Реализация слоя серверной логики и клиентского представления

Перед созданием логики внутри приложения Spring, необходимо прописать настройки проекта: подключение к базе данных, подключение к почтовому серверу и другие дополнительные настройки в файле application.properties, в том числе и настройки для Redis. Фреймворк будет автоматически подключаться к базе данных, и сохранять в ней сессии пользователей.

Далее можно рассмотреть, процесс обработки запросов, приходящих на сервер. Каждый раз, когда пользователь отправляет какую-либо форму в web-браузере, запрос отправляется на сервер. Spring MVC использует подход, называемый front-controller. В роли главного сервлета или front-controller, принимающего все входящие запросы выступает DispatcherServlet, его работа состоит в том, чтобы послать запрос на соответствующий Spring MVC контроллер. Он пользуется помощью обработчиков, называемых handler mappings которые помогает ему определить какой именно контроллер нужно вызывать. После чего сам контроллер начинает отработку [16]. Детальную схему обработки запроса можно увидеть на рисунке 4.2.1:

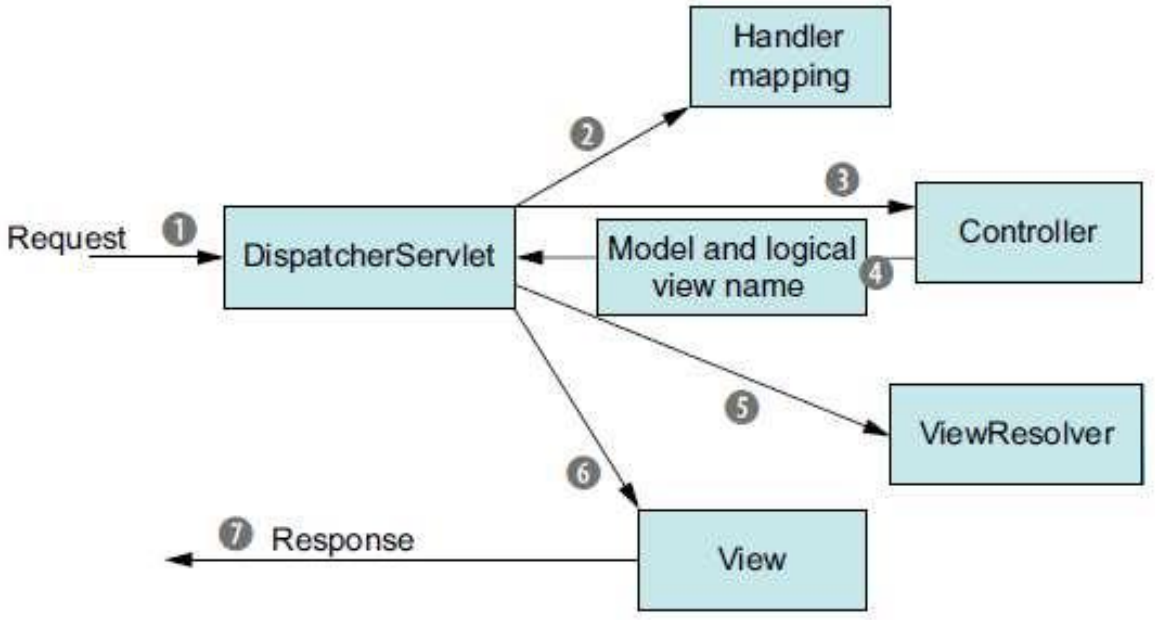

Рисунок 4.2.1 – Схема обработки запроса Spring MVC

Для обработки запросов, в приложении социальной сети используются следующие контроллеры:

- Registration Controller
- User Controller
- Message Controller

Первый контроллер обрабатывает запросы, которые приходят после регистрации, и переходу по индивидуальной ссылке для кода активации. Второй контроллер занимается обработкой запросов, которые приходят при изменении пользовательских данных, просмотров подписок пользователя, а также методы, доступные только суперпользователю, а именно просмотр всех пользователей и изменение их ролей. Третий контроллер занимается обработкой запросов, которые приходят при попытке добавить новое сообщение, изменить какое-либо из сообщений, и проставление отметок «Нравится» для сообщений.

Хорошо спроектированный контроллер делегирует работу по выполнению запроса на какой-либо сервисный метод, содержащий бизнес-логику. Для каждой из модели удобнее будет создать отдельные сервисы, в которых будет реализована вся бизнес-логика [17]. Это следующие классы:

- User Service
- Message Service

В пользовательском сервисе реализована бизнес-логика по работе с сущностью пользователя. В нем присутствуют методы по добавлению и обновлению пользователей, совершению подписок, а также отказа от подписки, активации самого пользователя и отправки письма на почту. В сервисе сообщений, реализована логика их фильтрации по определённому параметру, а также возможность вывода сообщений конкретного пользователя. Работа сервисов возможна в связке с репозиториями – интерфейсами, которые мы определяем для доступа к данным. Запросы создаются автоматически, основываясь на именах методов. Также можно

прописать запрос вручную, поскольку иногда нужна получить данные, не по конкретному имени заданному в коде [18]. Запросы для репозиториев, где в названии метода указано их назначение можно увидеть на рисунках 4.2.2 – 4.2.3:

```
public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
    User findByActivationCode(String code);
}
```

Рисунок 4.2.2 – Запросы для сущности пользователя

```
public interface MessageRepository extends CrudRepository<Message, Long> {
    @Query("select new com.example.socialnetwork.entities.dto.MessageDto(" +
            "  m," +
            " count(ml)," +
            " sum(case when ml = :user then 1 else 0 end) > 0" +
            ") " +
            "from Message m left join m.likes ml " +
            "group by m")
    Page<MessageDto> findAll(Pageable pageable, @Param("user") User user);

    @Query("select new com.example.socialnetwork.entities.dto.MessageDto(" +
            "  m," +
            " count(ml)," +
            " sum(case when ml = :user then 1 else 0 end) > 0" +
            ") " +
            "from Message m left join m.likes ml " +
            "where m.tag = :tag " +
            "group by m")
    Page<MessageDto> findByTag(@Param("tag") String tag, Pageable pageable, @Param("user") User user);


    @Query("select new com.example.socialnetwork.entities.dto.MessageDto(" +
            "  m," +
            " count(ml)," +
            " sum(case when ml = :author then 1 else 0 end) > 0" +
            ") " +
            "from Message m left join m.likes ml " +
            "where m.author = :author " +
            "group by m")
    Page<MessageDto> findByUser(Pageable pageable, @Param("author") User author);

}
```

Рисунок 4.2.3 – Запросы для сущности сообщения

После выполнения бизнес-логики пользователю нужно показать какую-то информацию как результат. Эта информация обычно храниться в модели. Однако информация перед показом должна пройти обработку, чтобы принять

удобоваримый вид, например HTML. Для этого контроллер посылает модель и название представления обратно в Dispatcher Servlet. Dispatcher Servlet посылает запрос по view resolver, чтобы понять, какую действительно HTML-страницу нужно показать пользователю. Для того чтобы создать данную HTML-страницу, необходимо использовать выбранный шаблонизатор FreeMarker, благодаря которому, можно брать данные из переданной контроллером модели, и подставлять их в виде гипертекста. Данный шаблонизатор удобен тем, что можно использовать одну HTML-страницу, и разбить ее содержимое на отдельные компоненты. Каждый компонент будет загружаться в теле страницы в зависимости от переданного пользователем запроса, делая страницу динамически изменяемой, реализую концепцию «Одностраничное приложение» [19].

В результате обработки пользовательских запросов, проведения бизнес-логики, сохранения и взятия необходимой информации из базы данных, наглядно данную схему можно увидеть на рисунке 4.2.4:

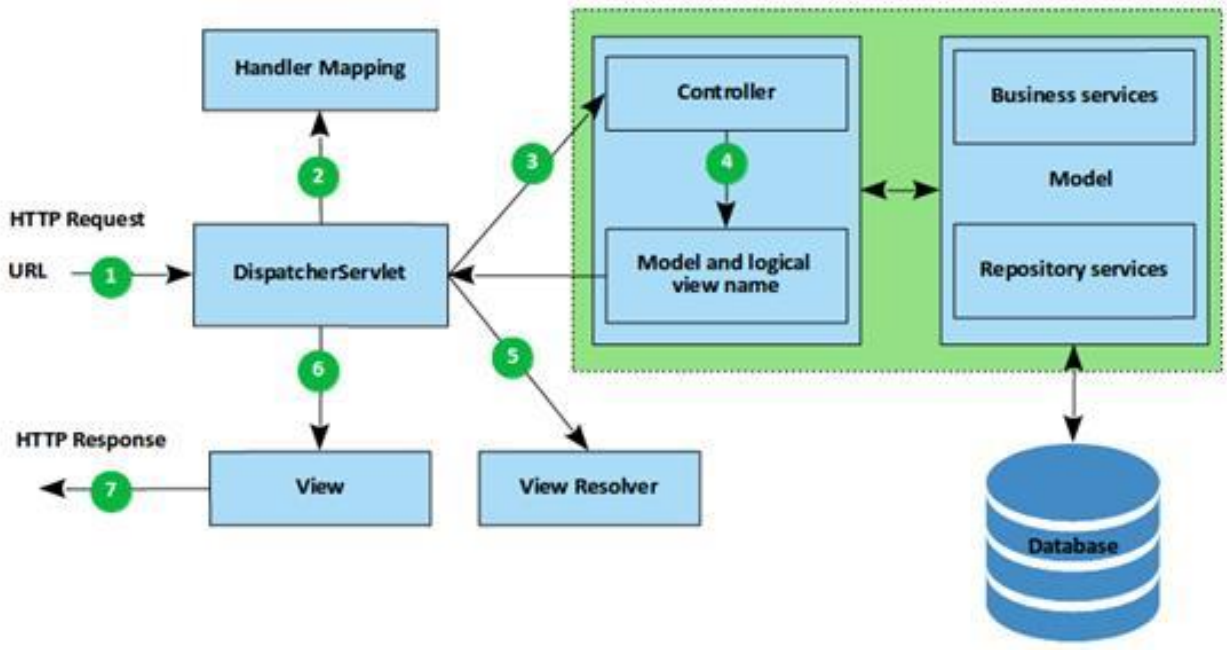

Рисунок 4.2.4 – Обработка пользовательского запроса

Итоговый программный код снабжен комментариями, для большего понимания логики внутри приложения. С ним можно ознакомиться на сохраненном репозитории [20], со всеми видимыми этапами разработки, на сервисе GitHub [21].

4.3 Тестирование и запуск разработанного интернет-ресурса

Перед запуском разработанного интернет-ресурса, необходимо сначала протестировать его. Тестирование позволит убедиться в том, что написанный код полностью обрабатывает приходящие запросы и выдает ожидаемый для нас результат. Для тестирования реализованного кода, было выбрано интеграционное тестирование. Это вид тестирования, при котором отдельные программные модули объединяются и тестируются вместе. Для тестирования кода в Java, используется фреймворк JUint. В приложении были протестированы основные компоненты, работа которых необходима для пользователей социальной сети.

Первый класс тестов содержит в себе методы для тестирования входа на главную страницу, регистрацию пользователя, его авторизацию, и проверку на то, что после выполнения входа пользователь был отправлен на страницу с сообщениями в приложении.

Второй класс тестов, содержит в себе методы для тестирования отправки сообщений, проверка присутствия сообщения в тексте, проверка фильтрации, проверка наличия текста и тэгов в сообщении.

Предварительно перед тестированием, в базу данных был добавлен тестовый пользователь, от имени которого и проводились описанные выше операции в тестировании. С кодом тестов можно также ознакомиться на сохраненном репозитории.

Итоговый результат тестирования можно увидеть на рисунке 4.3.1:

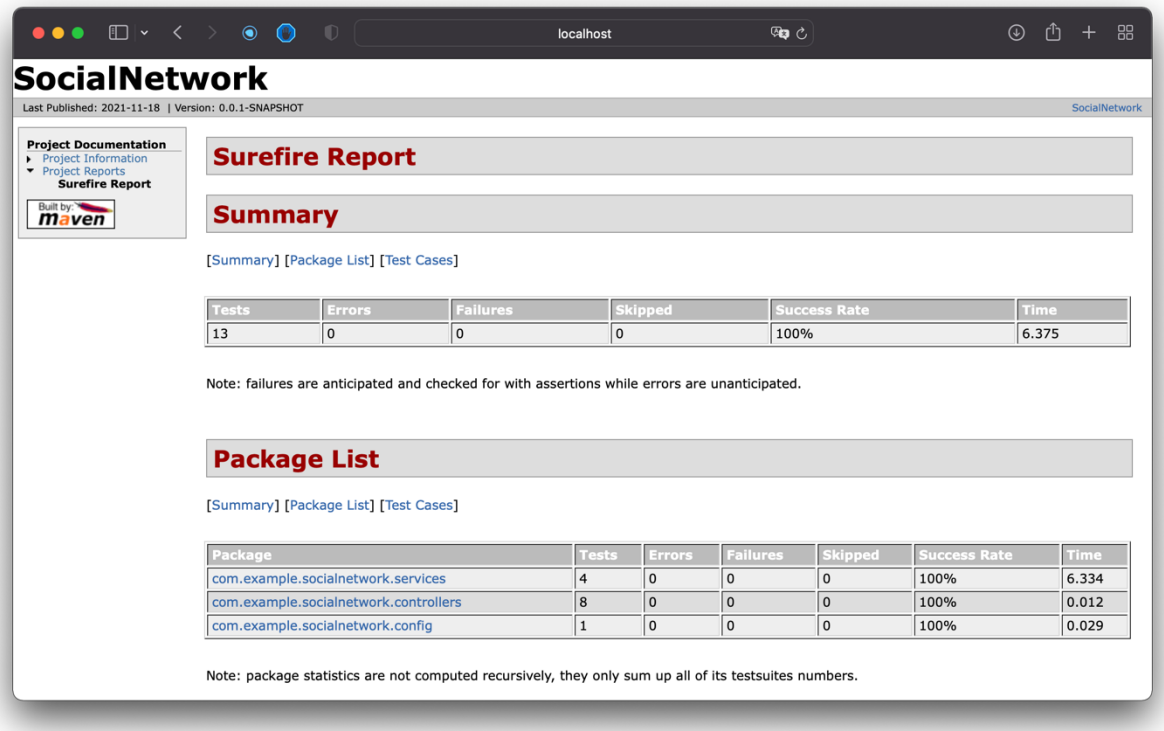

Рисунок 4.3.1 – Результаты тестирования

Далее на локальной машине был произведен запуск разработанного приложения, с регистрацией нового пользователя и добавлением новых сообщений от администратора и нового пользователя. Были произведены подписки пользователей друг на друга и проставления отметок «Нравится» для сообщений.

В результате с клиентской стороны функционал реализованного веб-приложения имеет вид, представленный на рисунке 4.3.2:

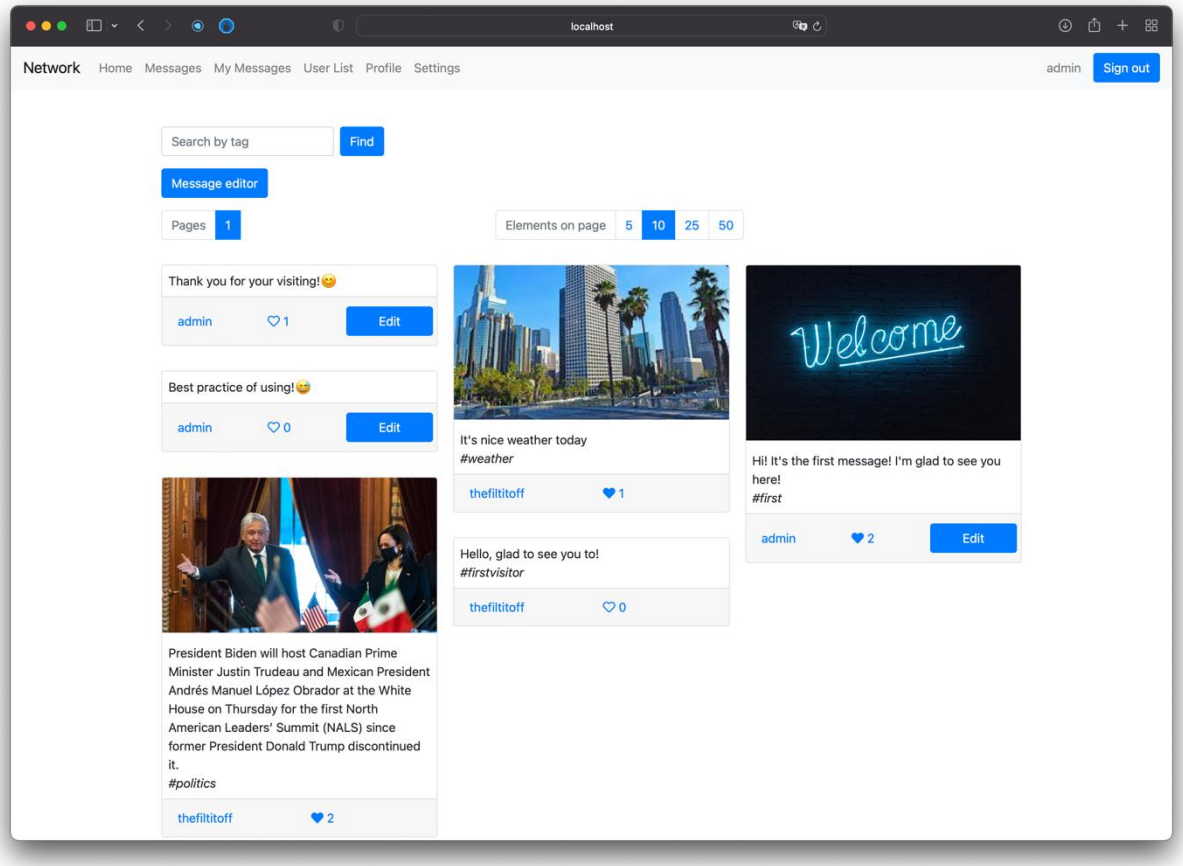

Рисунок 4.3.2 – Итоговый результат страницы

4.4 Внедрение разработанного интернет-ресурса

Последним этапом в цикле разработки ПО, является внедрение и сопровождения созданного приложения. Ключевой целью внедрения является постепенное выявление не обнаруженных ранее ошибок и недочетов кода. Для развертывания разработанного приложения был выбран сервис Heroku [22]. Данный сервис предоставляет платформу в качестве службы для развертывания приложений различных технологических стеков, в том числе и Java. Он заботится обо всех вложенных аспектах развертывания, инфраструктуры, масштабирования, обновления и безопасности, что позволяет в основном сосредоточиться только на логике приложения.

В Heroku приложение можно развернуть различными способами, мною был выбран наиболее простой и уже связанный с репозиторием исходного кода приложения.

Данный сервис позволяет подключить к сервису GitHub, и начать автоматически собирать приложение. Сервис позволяет бесплатно добавить к приложению необходимы для него компонент. Данным компонентом в нашем случае будет являться СУБД, которая необходима для корректной работы приложения.

Итоговый вариант приложения, развернутого на сервисе Heroku, можно увидеть на рисунке 4.4.1

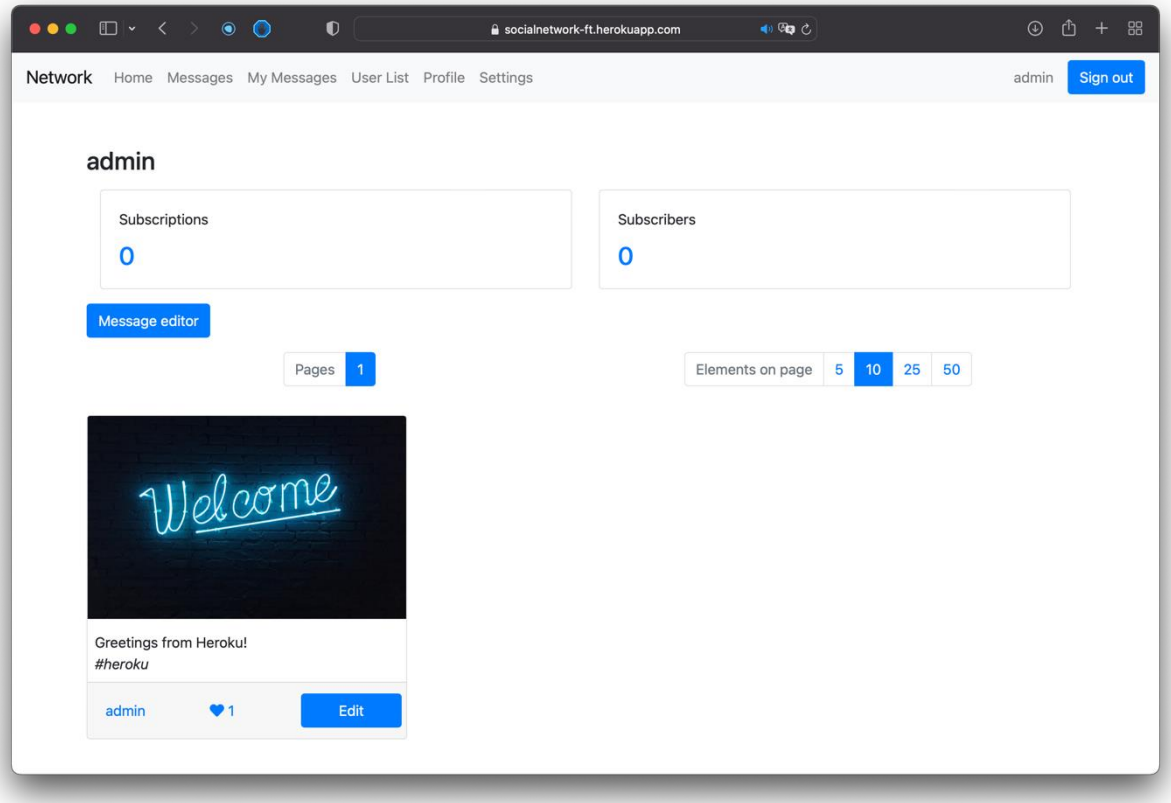

Рисунок 4.4.1 – Итоговый результат развернутого приложения

Приложение на сервисе полностью работоспособно, выполняет все действия, произведенные при запуске локально [23].

На данном сервисе, при длительном периоде, когда приложение не было использовано, может длительное время загружаться, поскольку сервис, в целях экономии ресурсов отправляет приложения в режим сна.

На сервис загружена первая версия приложения, которая в дальнейшем будет дополнятся новым функционалом и улучшениями.

Выводы к четвертому разделу

В результате анализа требований, предъявляемых к теме данного web-приложения, рассмотренных технологий, а также разработанной архитектуре, было разработано приложение, отвечающее поставленным требованиям. В данной главе был рассмотрен полный цикл разработки интернет-ресурса согласно варианту курсовой работы, от проектирования схемы базы данных, до развертывания приложения на сервисе.

Заключение

В ходе выполнения данной курсовой работы была реализована серверная часть web-приложения «Социальная сеть».

Спроектированное приложение дает возможность людям без труда, делиться информацией в режиме онлайн, рассказывать о своих интересах, общаться с интересными людьми и подписываться на них.

Данная тема выбрана для дальнейшего улучшения функционала уже созданного web-приложения при дальнейшем освоении новых навыков в уже выбранных технологиях для разработки.

Создание данного интернет-ресурса позволило закрепить знания и навыки, полученные в ходе изучения таких дисциплин, как: «Разработка серверных частей интернет-ресурсов» [24], «Архитектура клиент-серверных приложений» [25], «Интерфейсы прикладного программирования» [26], а также дисциплин «Разработка баз данных» [27] и «Тестирование и верификация сервисного ПО» [28].

Написание данной курсовой работы позволило научиться разрабатывать, проектировать и тестировать приложение, актуальные в современном мире.

Список использованных источников

1. Twitter – социальная сеть [Электронный ресурс]: Режим доступа: https://twitter.com/, свободный (Дата последнего обращения: 01.11.2021);
2. Facebook – социальная сеть [Электронный ресурс]: Режим доступа: https://www.facebook.com/, свободный (Дата последнего обращения: 01.11.2021);
3. IntelliJ IDEA – официальная документация [Электронный ресурс]: Режим доступа: https://www.jetbrains.com/idea/features/, свободный (Дата последнего обращения: 02.11.2021);
4. Язык программирования Java – официальная документация [Электронный ресурс]: Режим доступа: https://docs.oracle.com/en/java/, свободный (Дата последнего обращения: 02.11.2021);
5. Spring Framework - официальная документация [Электронный ресурс]: Режим доступа: https://spring.io, свободный (Дата последнего обращения: 02.11.2021);
6. PostgreSQL - официальная документация [Электронный ресурс]: Режим доступа: https://postgrespro.ru/docs/postgresql, свободный (Дата последнего обращения: 02.11.2021);
7. Redis – официальная документация [Электронный ресурс]: Режим доступа: https://redis.io, свободный (Дата последнего обращения: 03.11.2021);
8. JavaStudy [Электронный ресурс]: Режим доступа: https://javastudy.ru/spring-mvc/spring-mvc-basic/, свободный (Дата последнего обращения: 12.11.2021);
9. Мартин, Р. Чистая архитектура. Искусство разработки программного обеспечения / Р. Мартин. — СПб. : Питер, 2021. — 352 с.
10. Bootstrap - официальная документация [Электронный ресурс]: Режим доступа: https://getbootstrap.com/docs/5.0/, свободный (Дата последнего обращения: 02.11.2021);

11. Apache FreeMarker – официальная документация [Электронный ресурс]: Режим доступа: https://freemarker.apache.org, свободный (Дата последнего обращения: 02.11.2021);
12. SkillBox [Электронный ресурс]: Режим доступа: https://skillbox.ru/media/code/chto_takoe_mvc_bazovye_kontseptsii_i_primer_prilozheniya/, свободный (Дата последнего обращения: 12.11.2021);
13. SwiftBook [Электронный ресурс]: Режим доступа: https://swiftbook.ru/post/tutorials/common-design-patterns-and-app-architectures-for-android/, свободный (Дата последнего обращения: 12.11.2021);
14. Васильев, А.Н. Программирование для Java для начинающих/ А.Н. Васильев. — СПб. : Питер, 2021. — 704 с.
15. Дино, Э. Разработка современных веб-приложений: анализ предметных областей и технологий/ Э. Дино. — СПб. : Питер, 2021. — 464 с.
16. IntechCore [Электронный ресурс]: Режим доступа: https://ru.intechcore.com/stages-software-development/, свободный (Дата последнего обращения: 12.11.2021);
17. Харроп Р., Шефер К., Козмина Ю. Spring 5 для профессионалов/Харроп Р., Шефер К., Козмина Ю. — Москва, 2021. — 1120 с.
18. Шилдт Г. Java. Полное руководство/Шилдт Г. – Москва, 2021. – 1488 с.
19. Диков А. В. Клиентские технологии веб-дизайна. HTML5 и CSS3 [Электронный ресурс]: учебное пособие. - Санкт-Петербург: Лань, 2019. - 188 с. – Режим доступа: https://e.lanbook.com/book/122174
20. Исходный код проекта [Электронный ресурс]: Режим доступа: https://github.com/filtitov2001/SocialNetwork, свободный
21. GitHub [Электронный ресурс]: Режим доступа: https://github.com/, свободный (Дата последнего обращения: 17.11.2021);
22. Heroku – официальная документация [Электронный ресурс]: Режим доступа: https://heroku.com, свободный (Дата последнего обращения: 14.11.2021);

23. Развернутое приложение на сервисе Heroku [Электронный ресурс]: Режим доступа: https://socialnetwork-ft.herokuapp.com, свободный
24. Куликов А.А. Разработка серверных частей интернет-ресурсов [Электронный ресурс]: лекции - РТУ МИРЭА, Москва, 2021/2022 уч./год. (Дата последнего обращения: 12.11.2021);
25. Алпатов А.Н. Архитектура клиент-серверных приложений [Электронный ресурс]: лекции - РТУ МИРЭА, Москва, 2021/2022 уч./год. (Дата последнего обращения: 13.11.2021);
26. Алпатов А.Н. Интерфейсы прикладного программирования [Электронный ресурс]: лекции - РТУ МИРЭА, Москва, 2021/2022 уч./год. (Дата последнего обращения: 14.11.2021);
27. Богомольная Г.В. Разработка баз данных [Электронный ресурс]: лекции - РТУ МИРЭА, Москва, 2021/2022 уч./год. (Дата последнего обращения: 10.11.2021);
28. Чернов Е.А. Тестирование и верификация сервисного ПО [Электронный ресурс]: лекции - РТУ МИРЭА, Москва, 2021/2022 уч./год. (Дата последнего обращения: 12.11.2021);