Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И УПРАВЛЕНИЕ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ, ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ НА ТЕМУ:

Создание социальной сети

Студент группы ИУК4-52Б _____ Е.В. Губин
(подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы _____ С.А. Глебов
(подпись, дата) (И.О. Фамилия)

Калуга, 2024

Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой **ИУК4**
_____(_Ю.Е. Гагарин_)
«_9_»____сентября____2024г.

З А Д А Н И Е
на выполнение курсового проекта

по дисциплине ***Кроссплатформенная разработка программного обеспечения***

Студент группы ***ИУК4-52Б Губин Егор Вячеславович***
(фамилия, имя, отчество)
Тема курсовой работы ***Создание социальной сети***

Направленность КР ***учебная***
Источник тематики ***кафедра ИУК4***

Задание

Спроектировать структуру базы данных;
Разработать серверную часть;
Разработать клиентскую часть;

Оформление курсовой работы
Расчетно-пояснительная записка на 42 листах формата А4.
Перечень графического материала КР (плакаты, схемы, чертежи и т.п.):
− Use case диаграмма – 1 лист формата А3;
− Демонстрационный чертёже – 1 лист формата А3;
− Логическая ER-диаграмма сущностей – 1 лист формата А3;

Дата выдачи задания «_9_» сентября 2024 г.

Руководитель	9.09.2024	С.А. Глебов
	(подпись, дата)	(И.О. Фамилия)
Студент	9.09.2024	Е.В. Губин
	(подпись, дата)	(И.О. Фамилия)

Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного бюджетного образовательного учреждения высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

КАЛЕНДАРНЫЙ ПЛАН
на выполнение курсовой работы

по дисциплине ***Кроссплатформенная разработка программного обеспечения***

Студент группы ***ИУК4-52Б Губин Егор Вячеславович***
(фамилия, имя, отчество)

Тема курсовой работы Создание социальной сети

№	Наименование этапов	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Руководитель	Куратор
1	Задание на выполнение	1-я нед.			
2	Проектирование базы данных	3-я нед.			
3	Разработка серверной части	5-я нед.			
4	Разработка клиентской части	9-я нед.			
5	Защита	14-я нед.			

Студент _____ 9.09.2024г. Руководитель _____ 9.09.2024г.
(подпись, дата) (подпись, дата)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность темы курсового проекта обусловлена тем, что в настоящее время людям необходимо делиться информацией друг с другом, например, общение по работе, ведение личного блога, личное общение и многое другое.

Объектом курсового проекта является социальная сеть.

Предметом исследования курсового проекта является социальная коммуникация пользователей.

Целью проекта является разработка удобного сервиса, который сочетает в себе возможность общения и ведения блога о себе.

Для достижения поставленной цели решаются следующие задачи:

1. Выполнить анализ предметной области
2. Провести сравнительный анализ существующих аналогов
3. Определить оптимальную структуру системы
4. Осуществить выбор средств реализации программного продукта, соответствующего выбранной структуре.
5. Реализовать базу данных и программные компоненты системы
6. Осуществить тестирование компонентов
7. Разработать сопроводительную документацию

1. МЕТОДЫ И ИНСТРУМЕНТЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

1.1. Техническое задание

Название серивиса

Настоящее Техническое задание определяет требования и порядок создания сервиса для коммуникации и ведения личного блога «ChatBox». Название выбрано в связи с идеей «общение в одном месте».

Основания для разработки

Прежде всего люди ищут удобный способ для общения, обмена контентом или совместной работы.

Так же стоит отметить, что на рынке отсутствует социальная сеть, удовлетворяющая потребности определенной аудитории (например, профессионалов, художников, студентов). Данная аудитория нацелена на ведение блога о себе, своих произведениях или достижениях, а так же на обмен знаниями или ведение совместной работы.

Многим пользователям, которые пользуются аналогами, не нравится часто появляющаяся реклама.

В связи с вышеперечисленным необходимо создать сервис, который устраняет недостатки аналогов и предоставляет удобный формат для обмена информацией.

Исполнитель

Исполнителем проекта является студент Калужского филиала МГТУ им. Н. Э. Баумана, факультета ИУК, группы ИУК4-52Б, Губин Егор Вячеславович.

Краткая характеристика области применения

Разрабатываемая система предназначена для применения в области организации взаимодействия между пользователями, обмена информацией, ведения личных блогов. Она предоставляет инструменты для создания контента и общения в личных чатах.

Система ориентирована на удовлетворение потребностей аудитории в области личной и профессиональной коммуникации, распространения идей и

поиска единомышленников. Она может быть использована как платформа для социального взаимодействия, продвижения контента и формирования пользовательских сетей.

Целевая аудитория

Пользователи сети Интернет, интересующиеся социальными взаимодействиями, ведением блогов и обменом контентом.

Назначение системы

Разрабатываемая система должна обеспечивать удобное взаимодействие пользователей между собой.

Цели создания системы

Целью создания системы является предоставление удобной и функциональной платформы для взаимодействия пользователей, обмена информацией и ведения личных блогов. Система должна упростить процесс общения, стимулировать создание и распространение пользовательского контента.

Кроме того, система ориентирована на создание интуитивно понятной среды, где пользователи смогут делиться опытом, находить единомышленников.

Плановые сроки начала и окончания работы по созданию системы

Планируемые сроки начала и окончания работы над проектом: 09.09.2024 – 07.12.2024.

Требования к веб-приложению

Термины:

- Личная информация пользователя — ФИО, ник и почта пользователя.
- Ник — уникальное имя пользователя, начинается с символа «@». Его придумывает сам пользователь во время регистрации. Необходим для упрощения поиска пользователей.
- Пост — запись, которую делает пользователь; единица личного блога. Пост состоит из шапки, контента, комментариев и футера. Шапка содержит в себе аватар и ФИО автора поста и время (или дата или время и дата) создания поста. Контент — описание поста, его смысл, что хотел донести автор до аудитории. К контенту можно добавить фотографии, которые организованы в виде слайдера. Далее, в блоке комментариев, можно обозревать комментарии к посту или оставить свой комментарий. В футере есть возможность оставить реакцию на пост или сделать репост (копия поста) в свой блог с возможностью добавления своего контента. Авторизованный пользовтель не может сделать репост своего же поста. Если пост был сделан авторизованным пользователем, то он может удалить этот пост.
- Комментарий — отзыв или другое мнение о посте. Состоит из контента, времени (или даты или даты и времени) создания, аватара и ФИО автора комментария. Если комментарий был оставлен авторизованным пользователем или на его посте, то комментарий можно удалить.
- Аватар — картинка, которую ставит пользователь в качестве своей личной. При клике на любой аватар в любом месте сервиса необходимо переходить на личный блог этогот пользователя. Авторизованный пользователь может изменить аватар в разделе своего личного блога.
- Чат — место общения авторизованного пользователя с другим пользователем.

- Сообщение — единица общения. Представляет из себя контент, время (или дату или время и дату) отправки. Если отправителем является авторизованный пользователь, то сообщение можно удалить. Добавить возможность пересылки сообщения в этот же чат. Сообщение можно создать или переслать и отправить в открытом чате. Если сообщение приходит от любого пользователя, то отобразить это в виде уведомления, в котором есть контент сообщения, аватар и ФИО пользователя.
- Прочтение сообщения — событие, при котором авторизованный пользователь увидел сообщение в открытом чате.
- Подписчик/подписка — отношение между двумя пользователями, при котором один подписывается на другого. В этом случае, к примеру, первый становится подписчиком для второго. Для первого же второй является подпиской. Второй может добавить в друзья подписчика, а первый — отписаться от второго. Если второй добавит в друзья первого, то образуется отношение Дружба.
- Дружба — если между двумя подписчиками отношение — дружба, то каждый из них может удалить второго из друзей. В этом случае обратный процесс — второй становится подписчиком на того, кто удалил из друзей.
- Карточка пользователя — представляет из себя аватар и ФИО пользователя и кнопку взависимости от отношения между авторизованным пользователем и пользователем на карточке: subscribe, unsubscribe, add friend, delete friend. Через карточку пользователя можно перейти в чат с этим пользователем путем нажатия на специальную кнопку.

Реализация:

При входе в систему осуществить двухфакторную аутентификацию через почту с механизмом JWT авторизации. Двухфакторная аутентификация подразумевает отправку проверочного кода на почту и проверку этого кода от пользователя.

В случае, если пользователь не имеет аккаунта или не авторизован, то ему предлагается:

- Создать новый аккаунт. При создании нового аккаунта указывается личная информация пользователя, пароль и подтверждение пароля. В случае валидности данных создаётся аккаунт, после чего последует вход в систему.
- Войти в систему. Осуществляется вход в систему на основе почты и пароля пользователя.

Если пользователь авторизован, то сервис должен состоять из следующих разделов:

1. Profile - этот раздел представляет из себя личный блог и состоит из двух блоков: личная информация (здесь дополнительно отобразить аватар, количество друзей, подписчиков и подписок пользователя) и посты авторизованного пользователя. После блока личной информации, должны быть показаны посты пользователя.
2. Posts - в этом разделе собраны посты всех пользователей сервиса.
3. Chats — в этом разделе собраны все чаты. Карточка чата визуально представляют аватар и ФИО пользователя, последнее сообщение в этом чате и его время (или дата или дата и время). Если последнее сообщение было отправлено авторизованным пользователем и собеседник его не прочитал, то отметить его непрочитанным виде точки. Если последнее сообщение от собеседника и авторизованный пользователь его не прочитал, то отобразить чат непрочитанным в виде измения фона карточки чата и отображения количество непрочитанных сообщений. В

открытом чате отобразить в шапке ФИО и аватар пользователя и кнопку Back, а ниже все сообщения: слева одним цветом - от собеседника, справа другим цветом — от авторизованного пользователя. Непрочитанные сообщения от авторизованного пользователя отобразить частично прозрачными.

4. Friends — список друзей авторизованного пользователя в виде карточек пользователей этих друзей.
5. Subscribers — список подписчиков авторизованного пользователя в виде карточек пользователей этих друзей.
6. Subscribes — список подписок авторизованного пользователя в виде карточек пользователей этих друзей.

Требования к надежности

Программный продукт должен устойчиво функционировать и обеспечивать надежную защиту данных.

Надежное (устойчивое) функционирование системы должно быть обеспечено выполнением пользователем (заказчиком) совокупности организационно-технических мероприятий, перечень которых приведен ниже:

— организацией бесперебойного питания технических средств;

— использованием лицензионного программного обеспечения;

— регулярным выполнением требований ГОСТ 51188-98 (защита информации, испытание компьютера на наличие компьютерных вирусов);

Условия эксплуатации

- стабильное электропитание серверов
- поддержка современных веб-браузеров (Chrome, Firefox, Edge, Safari) версий не ниже текущих минус два
- подключение к интернету с минимальной скоростью 1 Мбит/с для корректной загрузки ленты, мультимедиа и сообщений.

- рекомендуемый объем оперативной памяти для комфортной работы на устройстве — 2 ГБ и выше.
- приложение ориентировано на использование в нормальных офисных или домашних условиях.
- защита от несанкционированного доступа и регулярное обновление системы безопасности.
- уровень доступности системы: не менее 99,5% времени работы в месяц.

Требования к защите информации и программ

Для защиты используется современная двухфакторная аутентификация через отправку проверочного кода на почту пользователя, технология JWT авторизации, а именно проверка валидности токена при каждом запросе к серверу, и хеширования некоторых данных при хранении в базе данных.

Требования к программной документации

Должны быть разработаны следующие программные документы:

1. Расчетно-пояснительная записка:

— Техническое задание;
— Научно-исследовательская часть;
— Проектная часть;
— Производственно-технологическая часть;
— Организационно-экономическая часть;
— Раздел охраны труда и экологии;

2. Графическая часть - 3 листа формата А3 включающие в себя:

— демонстрационные чертежи;
— use case диаграмма;
— логическая ER-диаграмму.

Стадии разработки

Техническое задание

1. Обоснование перспективности реализуемого проекта:

— постановка задачи;

— сбор базовых материалов;

— установка критериев системы;

— необходимость проведения исследовательских работ;

2. Исследовательская работа:

— выбор оптимальных методов решения поставленной задачи;

— определение требований к техническим средствам;

— обоснование практической возможности реализации данного проекта;

3. Разработка и утверждение технического задания:

— определение требований к проекту;

— определение стадий, этапов и сроков разработки проекта и документации на нее;

— согласование и утверждение технического задания;

Технический проект

1. Разработка технического проекта:

— определение формы представления входных и выходных данных;

— определение конфигурации технических средств;

2. Утверждение технического проекта:

— установка плана по разработке проекта;

— создание пояснительной записки;

— утверждение технического проекта;

Этапы разработки

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии проектирования программы должен быть выполнен этап выборки программного обеспеченья, системы управления базой данных,

библиотек для создания, этап проектирования системы в целом, разработка рабочей документации.

На стадии реализации производится разработка и тестирование спроектированной программы.

1.2. Анализ существующих аналогов

В настоящее время существуют аналоги, предоставляющие схожий функционал, такие как популярные социальные сети (например, Facebook, Instagram). Однако данные системы обладают избыточным набором функций, которые могут быть не востребованы значительной частью пользователей, усложняя процесс взаимодействия и освоения платформы.

Кроме того, такие аналоги часто требуют значительных аппаратных ресурсов, что ограничивает их доступность для пользователей с менее производительными устройствами. Также некоторые платформы недостаточно адаптированы под локальные особенности и специфические потребности аудитории, что снижает их эффективность для целевой группы проекта.

Разрабатываемая система направлена на создание упрощенной и интуитивно понятной платформы, ориентированной на удовлетворение основных потребностей пользователей без избыточного функционала.

1.3. Обоснование выбора ОС для разработки

Операционная система Linux является оптимальным выбором для разработки серверной части социальной сети и может использоваться как основная платформа для создания и развертывания системы. Вот ключевые причины:

- Linux известен своей стабильностью и предсказуемым поведением даже при высоких нагрузках, что критично для серверов социальной сети, работающих 24/7.
- Linux оптимизирован для работы с большим количеством параллельных соединений, что делает его подходящим для серверов, обрабатывающих тысячи или миллионы запросов в секунду.

- Возможности тонкой настройки ядра и параметров системы позволяют адаптировать ОС под конкретные нужды проекта, например, для улучшения обработки API-запросов или работы с базами данных.
- Большинство популярных технологий, используемых для разработки социальных сетей (PostgreSQL, Redis, Nginx, Docker, Kubernetes), оптимизированы и протестированы для работы на Linux.
- Linux имеет развитую систему управления безопасностью, включая гибкую настройку прав доступа и возможности для автоматического обновления системы.
- Linux является бесплатной ОС с открытым исходным кодом, что значительно снижает затраты на лицензирование.
- Упрощение автоматизацию разработки, тестирования, деплоя и масштабирования.

1.4. Обоснование выбора СУБД

Общие сведения

Разрабатываемая система предполагает хранение и обработку данных, следовательно, необходимо средство для эффективного и удобного выполнения этих задач. Наилучшим образом для этого подходят базы данных.

Различают три основных типа баз данных: иерархическую, сетевую и реляционную. В настоящее время иерархические и сетевые модели данных уступают реляционной. Такое положение обосновано рядом имеющихся у реляционной модели преимуществ, среди которых: простота и наглядность в сравнении с другими двумя моделями, удобство физической реализации на ЭВМ, применение строгих правил при проектировании базирующихся на математическом аппарате, независимость данных.

Для взаимодействия с базами данных используются системы управления базами данных (СУБД). Для разработки базы данных была выбрана СУБД PostrgreSQL.

СУБД PostgreSQL

Свободно распространяемая СУБД, относится к объектно-реляционному типу. PostgreSQL отличается тем, что предоставляет объектно-ориентированный функционал, в том числе обладает поддержкой концепта ACID (атомарность, непротиворечивость, изолированность, сохранность данных).

Преимущества:

— Полная SQL-совместимость.

— Расширяемость: PostgreSQL —можно программно расширить за счёт хранимых процедур.

— Объектно-ориентированность: PostgreSQL — не только реляционная, но и объектно-ориентированная СУБД.

Недостатки:

— Производительность: в простых операциях чтения PostgreSQL может уступать своим соперникам, таким как MySQL.

— Популярность: из-за своей сложности инструмент не очень популярен.

— Подходит больше для работы со сложными структурами данных и не менее сложными процедурами.

1.5. Обоснование выбора языков программирования

HTML5/TypeScript - являются сценарными языками программирования. Они несомненно обладают рядом преимуществ, например, кроссплатформенность, придание интерактивности веб-страницам. Поскольку разрабатываемое приложение является видом нативного приложения с тесным использованием функций ПО смартфорна (камеры), данный стандарт не подходит для поставленной задачи. TypeScript, в свою очередь, является строго типизированным языком, что делает приложение более надёжным и урощает разработку.

Веб-приложение имеет клиент-серверную структуру. Сервреная часть создана с помощью мощной библиотки ExpressTS, а клиентская с помощью библиотеки ReactTS.

Обоснование:

- ReactTS является библиотекой для создания SPA (single page application — ускоряет работу приложения за счёт того, что содержание всех страниц отоюражается на одной и той же странице, а не на разных). Выбор этой библиотеки заключается в создании переиспользуемых компонентов и использовании виртуальной копии DOM-дерева, что позволяет отображать все страницу целиком, а только её изменения.
- ExpressTS предоставляет лишь основные функции для создания веб-приложений, что делает его легким и гибким. Это позволяет разработчикам сосредоточиться на бизнес-логике приложения, а не на инфраструктуре. Простота API и небольшой объем кода для создания серверов делают Express отличным выбором для быстрых прототипов и приложений с низким порогом входа. ExpressTS позволяет использовать middleware для обработки HTTP-запросов, таких как парсинг JSON, проверка аутентификации.
- В совокупности сервера, основанные на этих библиотеках, легко соединить друг с другом для взаимодействия.

1.6. Обоснование выбора сред разработки

Visual Studio Code — текстовый редактор с кроссплатформенной поддержкой. Позволяет использовать множество рассширений, которые упрощают написание кода.

DBeaver — это универсальная платформа для работы с базами данных, которая поддерживает множество различных СУБД (систем управления базами данных).

1.7. Описание реализуемой архитектуры клиент-сервер

Разрабатываемое приложение использует архитектуру клиент-сервер для взаимодействия с удаленной базой данных. Эта архитектура представляет собой распределенную структуру, состоящую из двух основных компонентов: сервера, который обрабатывает запросы, и клиентов, инициирующих эти

запросы. Между сервером и клиентами устанавливается связь через определённый протокол.

В рамках системы сервер хранит базу данных, содержащую информацию о пользователях, постах, сообщениях, комментариях, реакциях, друзьях, подписках и подписчиках. Сервер принимает запросы от клиентов, обрабатывает их и возвращает соответствующие ответы. Клиенты, в свою очередь, формируют запросы для получения данных о различных объектах, которые затем отображаются пользователю.

1.8. Выбор средства для взаимодействия с базой данных

В JavaScript для работы с удаленной базой данных PostgreSQL одной из популярных библиотек является **pg** (node-postgres). Библиотека **pg** предоставляет API для взаимодействия с PostgreSQL, позволяя выполнять SQL-запросы, управлять подключениями и обрабатывать результаты.

Библиотека **pg** является универсальной и позволяет работать с PostgreSQL-сервером, абстрагируя детали подключения и работы с базой данных, используя промисы и асинхронные функции. Она предоставляет функциональность для выполнения различных типов SQL-запросов, работы с транзакциями и других операций с базой данных.

Конкретная реализация **pg** включает в себя возможности для работы с различными типами запросов, поддерживает пул подключений для повышения производительности и управления нагрузкой, а также использует асинхронные методы для эффективной работы в условиях многозадачности.

Выводы

Исходя из требований к разрабатываемой системе, анализа возможностей наиболее подходящих инструментов и их сравнения было принято решение использовать следующие решения:
В качестве ОС для разработки была выбрана **Linux**, что обусловлено её стабильностью, открытым исходным кодом, удобной поддержкой современных технологий и инструментов, а также предпочтениями разработчиков.

- для работы с базой данных была выбрана **PostgreSQL**, поскольку эта СУБД обладает высокой производительностью, расширяемостью и надежностью, а также хорошей поддержкой для сложных запросов и обработки больших объемов данных. В качестве инструмента для работы с PostgreSQL используется **Dbeaver**.
- Для реализации серверной части было решено использовать **Express** в связке с **TypeScript**, так как это позволяет эффективно создавать RESTful API с типизированными запросами и ответами, а также обеспечивает хорошую интеграцию с Node.js.
- В качестве среды разработки был выбран **Visual Studio Code**, поскольку это легкий, мощный и гибкий редактор с множеством расширений для TypeScript и React.
- Для разработки клиентской части приложения используется **React** с **TypeScript**, что позволяет создавать масштабируемые и легко поддерживаемые пользовательские интерфейсы, используя типизацию и компоненты.

2. ПРОЕКТИРОВАНИЕ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

2.1. Разработка структуры системы

Разрабатываемая система состоит из следующих компонентов:

База данных – хранит всю необходимую информацию о всех сущностях веб-приложения: посты, пользователи, комментарии, реакции, чаты, сообщения и картинки.

Серверная часть – представляет собой компьютер-сервер на базе ОС Linux, реализующий бизнес логику приложения. Перед тем, как принять запрос, с помощью middleware проверяется валидность токенов пользователя, если токены не валидны, то на клиент отправляется 401 ошибка. После проверки валидности токенов сервер принимает запрос, обращается к базе даных и отправляет ответ на сервер.

Клиентская часть – программа, на которой формируются запросы и отправляются на сервер с помощью HTTP-запросов посредством axios. Если в качестве ответа приходит 401 ошибка, то клиент автоматически посылает запрос на сервер на получение новой пары токенов (на основе refresh — токена).

2.2. Разработка базы данных

База данных состоит из следующих таблиц:

— пользователи;
— коды активации;
— refresh-токены;
— посты;
— дружба;
— подписчики;
— реакции;
— комментарии;
— чаты;
— сообщения;

— картинки пользователей;

— картинки постов.

Таблица 1 - «Пользователи»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
user_id	INT	PK	-
last_name	VARCHAR(50)	-	NOT NULL
first_name	VARCHAR(50)	-	NOT NULL
patronymic	VARCHAR(50)	-	-
email	TEXT	-	NOT NULL, UNIQUE
nickname	VARCHAR(50)	-	NOT NULL, UNIQUE
hashed_password	VARCHAR(255)	-	NOT NULL
is_online	BOOLEAN	-	DEFAULT NULL

Таблица 2 - «Активационные коды»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
activation_code_id	INT	PK	-
activation_code	INT	-	NOT NULL
user_id	INT	FK	NOT NULL, UNIQUE

Таблица 3 - «Refresh-токены»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
refresh_token_id	INT	PK	-
refresh_token	VARCHAR(255)	-	NOT NULL
expires_date	TIMESTAMP WITHOUT TIME ZONE	-	NOT NULL
user_id	INT	FK	NOT NULL, UNIQUE

Таблица 4 - «Посты»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
post_id	INT	PK	-
content	TEXT	-	-
publication_date_time	TIMESTAMP WITHOUT TIME ZONE	-	NOT NULL
children_post_id	INT	FK	DEFAULT NULL
post_author_id	INT	FK	NOT NULL

Таблица 5 - «Друзья»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
friendship_id	INT	PK	-
first_friend_id	INT	FK	NOT NULL
second_friend_id	INT	FK	NOT NULL

Таблица 6 - «Подписчики»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
subscriber_page_owner_id	INT	PK	-
subscriber_id	INT	FK	NOT NULL
page_owner_id	INT	FK	NOT NULL

Таблица 7 - «Реакции»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
reaction_id	INT	PK	-
post_id	INT	FK	NOT NULL
reaction_author_id	INT	FK	NOT NULL

Таблица 8 - «Комментарии»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
comment_id	INT	PK	-
content	TEXT	-	NOT NULL
comment_date_time	TIMESTAMP WITHOUT TIME ZONE	-	NOT NULL
post_id	INT	FK	NOT NULL
comment_author_id	INT	FK	NOT NULL

Таблица 9 - «Чаты»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
chat_id	INT	PK	-
first_user_id	INT	FK	NOT NULL
second_user_id	INT	FK	NOT NULL

Таблица 10 - «Сообщения»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
message_id	INT	PK	-
content	TEXT	-	NOT NULL
dispatch_date_time	TIMESTAMP WITHOUT TIME ZONE	-	NOT NULL
is_checked	BOOLEAN	--	NOT NULL, DEFAULT NULL
children_message_id	INT	FK	DEFAULT NULL
sender_id	INT	FK	NOT NULL
chat_id	INT	FK	NOT NULL

Таблица 11 - «Аватары»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
profile_image_id	INT	PK	-
date_time_publication	TIMESTAMP WITHOUT TIME ZONE	-	NOT NULL
image_data	BYTEA	-	NOT NULL
mime_type	VARCHAR(10)	-	NOT NULL
user_id	INT	FK	NOT NULL

Таблица 12 - «Картинки постов»

Наименование поля	**Тип данных**	**Признак ключа**	**Примечание**
post_image_id	INT	PK	-
image_data	BYTEA	-	NOT NULL
mime_type	VARCHAR(10)	-	NOT NULL
post_id	INT	FK	NOT NULL

2.3. Построение диаграммы вариантов использования веб-приложения

Подробное описание возможностей данного компонента системы удобно представить при помощи языка прецедентов. Прецедент представляет собой один из вариантов использования приложения. Совокупность же прецедентов помогает определить перечень действий для достижения необходимого результата. Разработанная диаграмма представлена на чертеже use case диаграммы.

2.4. Организация сетевого взаимодействия

В разрабатываемом сервисе представлена клиент-серверная архитектура. Сервер получает запросы, обрабатывает их, выполняет определенные действия и отправляет ответ на клиент. Клиент же, в свою очередь, формирует запросы на основе действий пользователя и отправляет их на сервер. Далее принимает ответ и, если необходимо, отображает результаты запросы пользователю.

Сервер

Вся суть организации сервера заключается в совокупности роутеров, контроллеров, сервисов и dto (data transfer object).

Роутеры отвечают на маршрутизацию запроса внутри сервера и за выполняемые действия. Каждый роутер привязывается к какой-либо сущности и определяет для него функционал: прописывается метод принимаемого запроса и url адрес для него. Вместе они должны организовывать уникальный маршрут, при запросе по которому происходит определнное действие, иначе сервер не будет точно знать, что какое конкретно действие производить над данными.

Болшую часть действий может выполнять только авторизованный пользователь. Для этого в библиотеке Express предусмотрено создание middleware, промежуточных слоёв, через которые проходят запросы. Перед тем как выполнить какое либо действие, запрос попадает в промежуточный слой, который проверяет авторизацию пользователя. Это происходит на основе валидации access токена. Если токен невалидный, то сервер посылает на клиент 401 статус код (пользователь не авторизован).

Блок схема промежуточного слоя, проверяющего валидность токена:

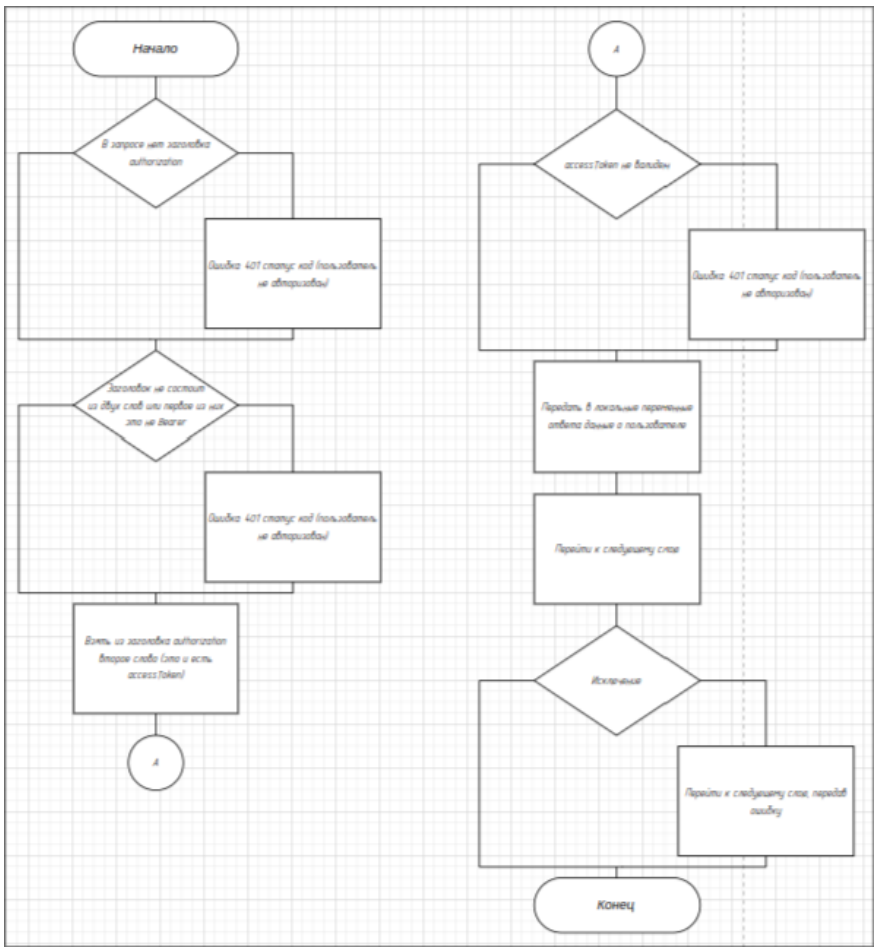

Рис. 1: Блок схема промежуточного слоя, проверяющего валидность токена пользователя

Если возникает ошибка при валидации токена пользователя, то в error middleare пробрасывается ошибка 401 (пользователь не авторизован). Error middleware принимает ошибки и отправляет их на сервер.

Данные с клиента передаются с помощью тела и/или параметров запроса. Если токен валидный, то роутер пропускает запрос дальше, в промежуточный слой валидации данных тела и/или параметров запроса. Результат валидации приходит в функцию обработки запроса. В этой функции проверяется наличие ошибок валидации: если ошибки есть, то они пробрасываются в error middleware.

Контроллеры хранят в себе методы для обработки запросов. Каждый контроллер привязывается к конкретной сущности.

Сервисы предоставляют функционал контроллерам и обеспечивают взаимодействие с базой данных.

Dto используется для передачи данных между слоями или компонентами приложения. Он играет важную роль в обеспечении структурированности, безопасности и удобства работы с данными.

Клиент

Клиент формирует запросы и принимает ответы. Какждый ответ и запрос обрабатываются перехватчиками (interceptor).

Перехватчик запроса добавляет к заголовкам запроса заголовок authorization, в которй передаётся строка из двух слов: Bearer и сам accessToken.

Перехватчик ответа обрабатывает 401 код ошибки: посылает новый запрос на сервер для обновления пары токенов (access и refresh токен). Далее после запроса refresh токен обновляется в куки клиента, а access токен сохраняется в локальном хранилище. После обновления пары токенов клиент повторяет запрос, в ответ на который пришел 401 статус код.

2.5. Двустороннее связывание клиентов.

Двустороннее связывание клиентов через сокеты — это ключевой механизм для создания приложений реального времени. Этот подход позволяет клиентам общаться друг с другом через сервер, обеспечивая моментальную передачу сообщений.

В разрабатываемом сервисе технология сокетов используется для общения в чате. Когда пользователи общаются, отправленные сообщения должны появляться не только у самого клиента, но и у клиента собеседника.

Клиенты подключаются к серверу с помощью протокола WebSocket. Это обеспечивает постоянное двустороннее соединение между клиентом и сервером.

Пользователь открывает чат, а сервер инициирует обработку подключения и регистрирует клиента по id сокета. Когда один клиент отправляет сообщение, оно пересылается на сервер через открытое соединение WebSocket. Сервер получает сообщение, определяет получателя и пересылает сообщение адресату. Адресат получает новое сообщение от собеседника так же по отрытому соединению

Выводы

В разработанном сервисе реализована клиент-серверная архитектура с четким разделением задач между клиентом и сервером, что обеспечивает гибкость, безопасность и простоту масштабирования.

3. ТЕСТИРОВАНИЕ И ИНТЕГРАЦИЯ КОМПОНЕНТОВ ПРОГРАММНОГО ПРОДУКТА

3.1. Тестирование сервиса

Тестирование сервиса производилось с использованием персонального компьютера на котором были запущены два клиента, сервер и база данных. Два клиента использовались для тестирования чатов, работы сокетов и проверки корректного отображения данных другого пользователя. Тестирование проводилось с большим количеством данных: отправка множества сообщений, создание множества постов и многое другое.

Требования к характеристикам системы для веб-приложения

- Операционная система: Windows 10/11, macOS (последние три версии), Linux (например, Ubuntu 20.04+).
- Процессор: двухъядерный (например, Intel Core i3 или эквивалент).
- Оперативная память: минимум 4 ГБ (8 ГБ рекомендуется).
- Хранилище: 500 МБ свободного пространства для кэша браузера

3.2. Руководство пользователя

При первом визите сервиса пользователю предлагается зарегистрироваться или войти под существующим аккаунтом:

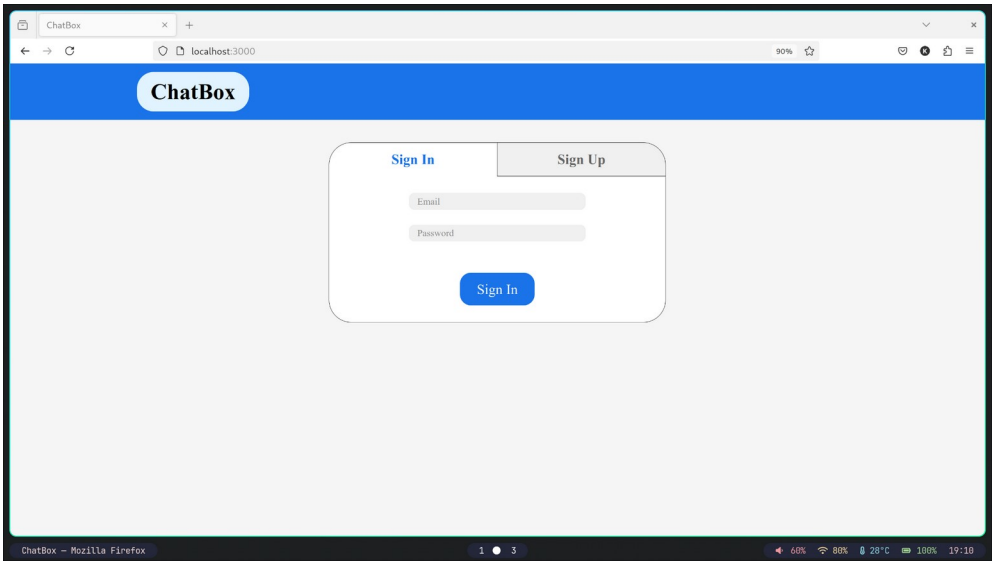

Рис. 2: Форма входа в аккаунт

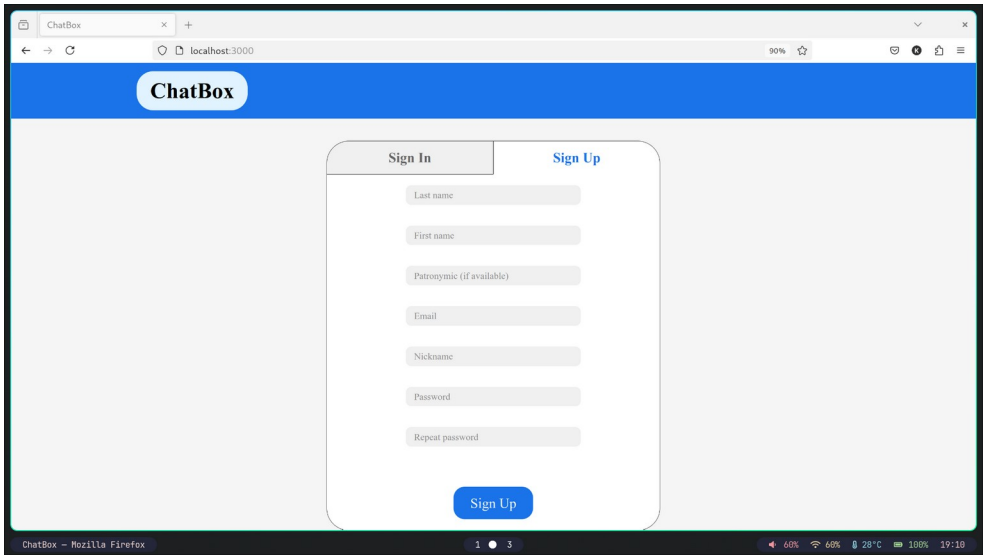

Рис. 3: Форма регистрации нового аккаунта

После регистрации или входа в существующий аккаунт (если введённые данные правильные) появляется окно ввода подтвержающего кода, который был отправлен на почту:

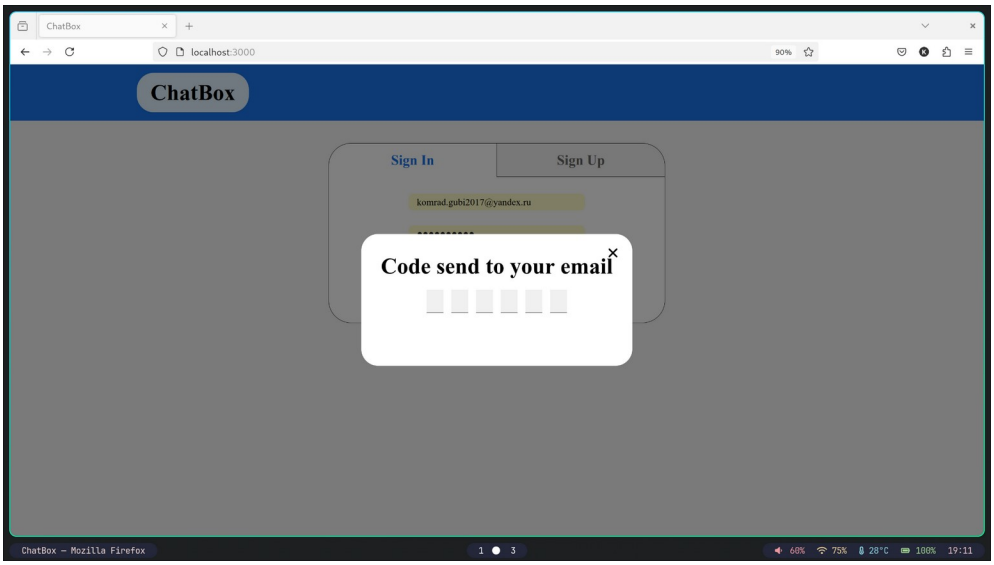

Рис. 4: Окно для ввода кода подтверждения

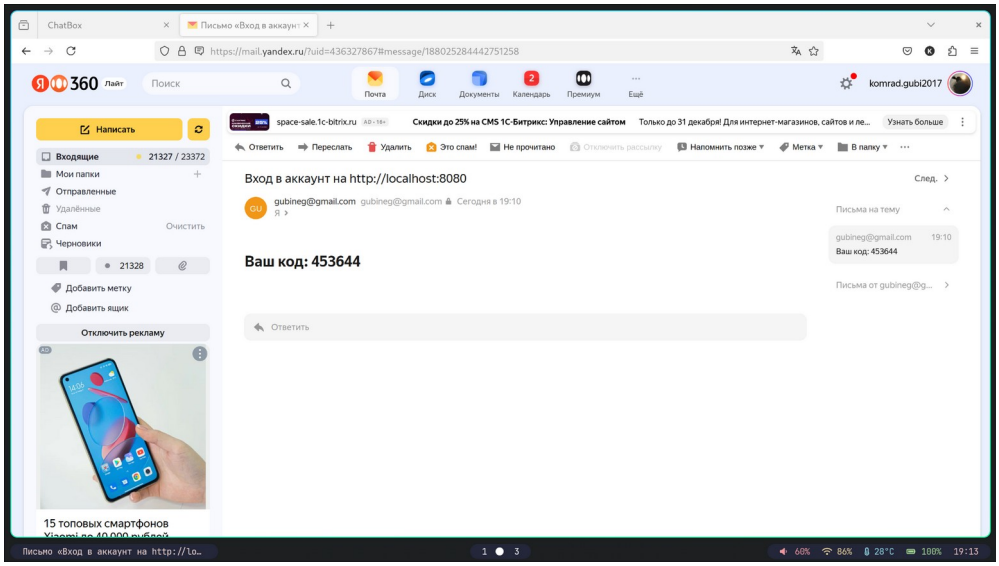

Рис. 5: Код, который пришел на почту

После успешного ввода кода появляется личный блог авторизованного пользователя:

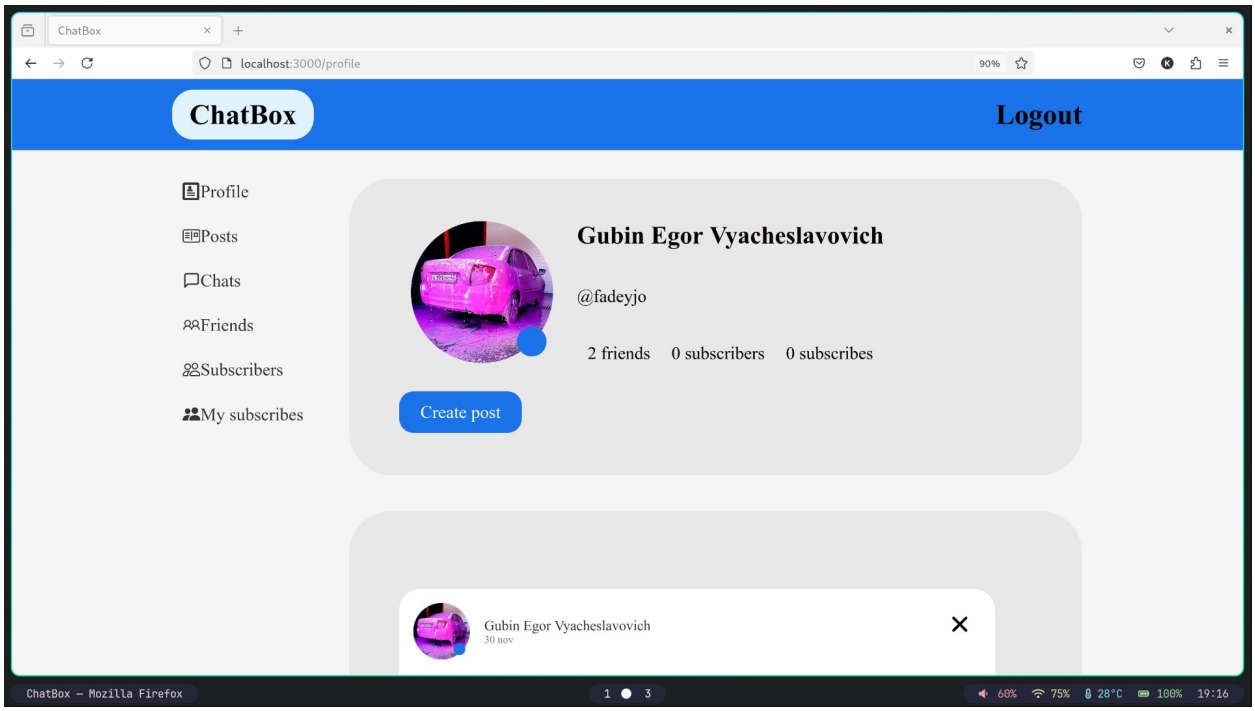

Рис. 6: Личный блог пользователя

Страница чатов:

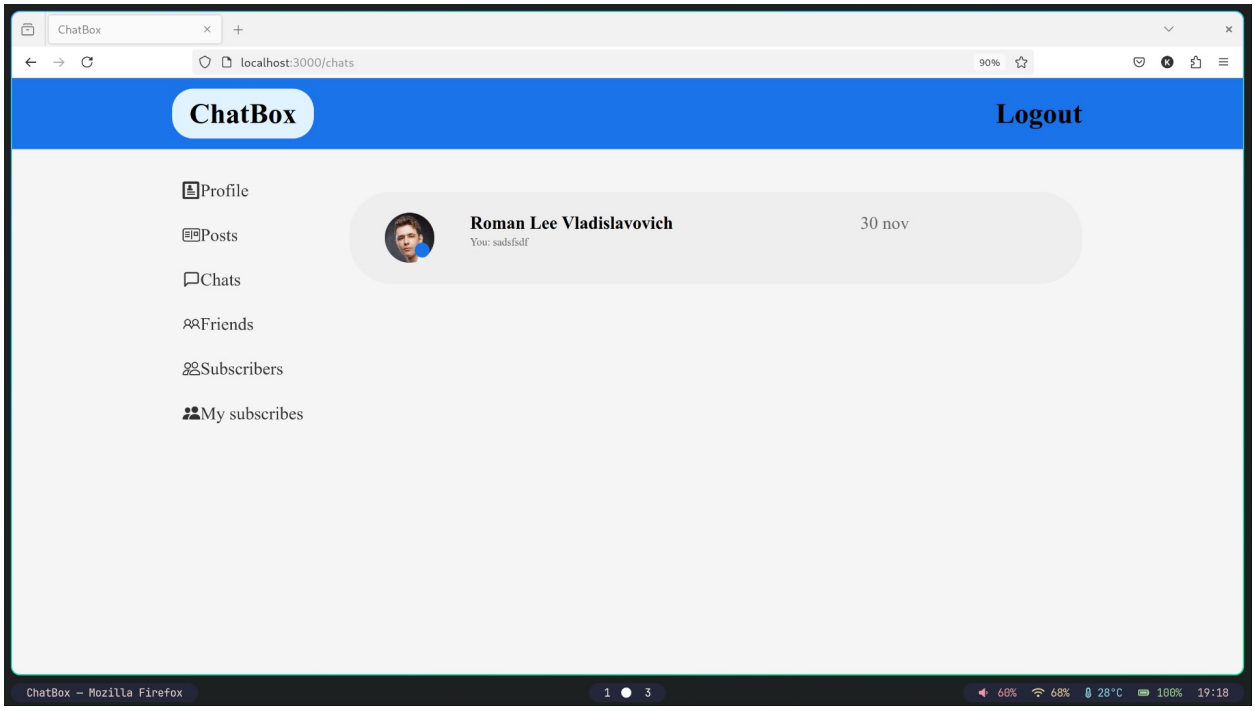

Рис. 7: Чаты

Страница постов:

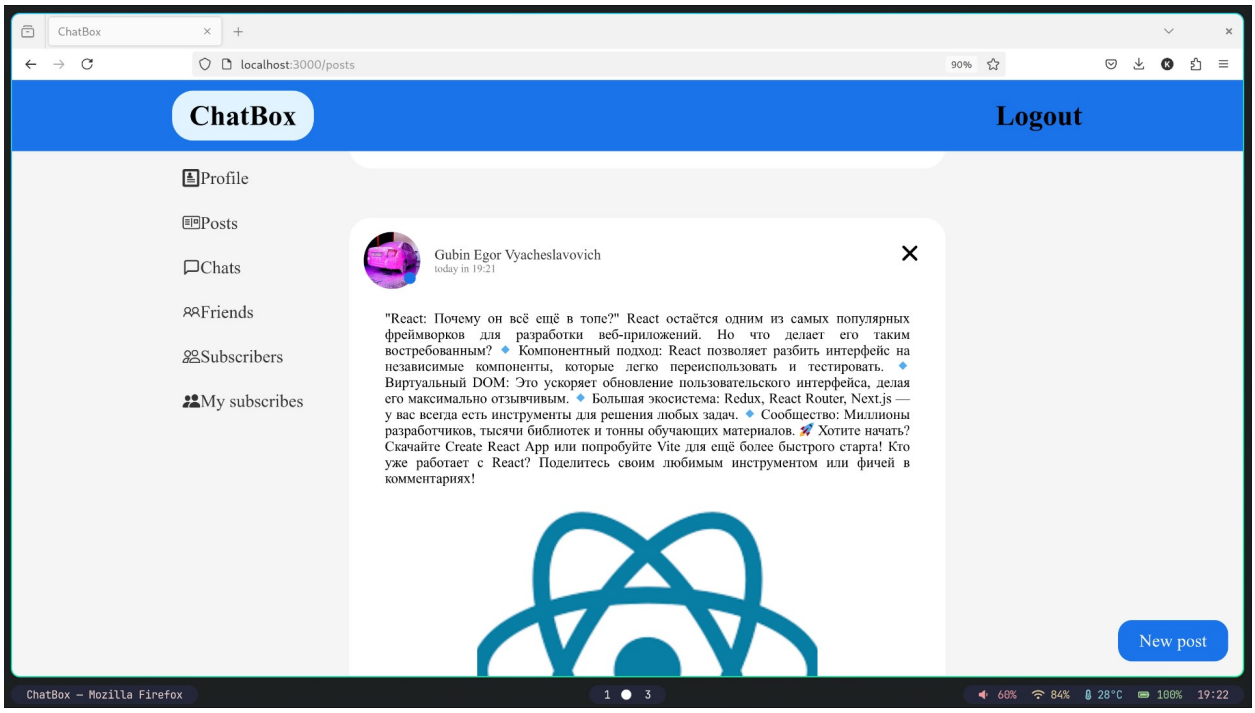

Рис. 8: Посты

Страница друзей:

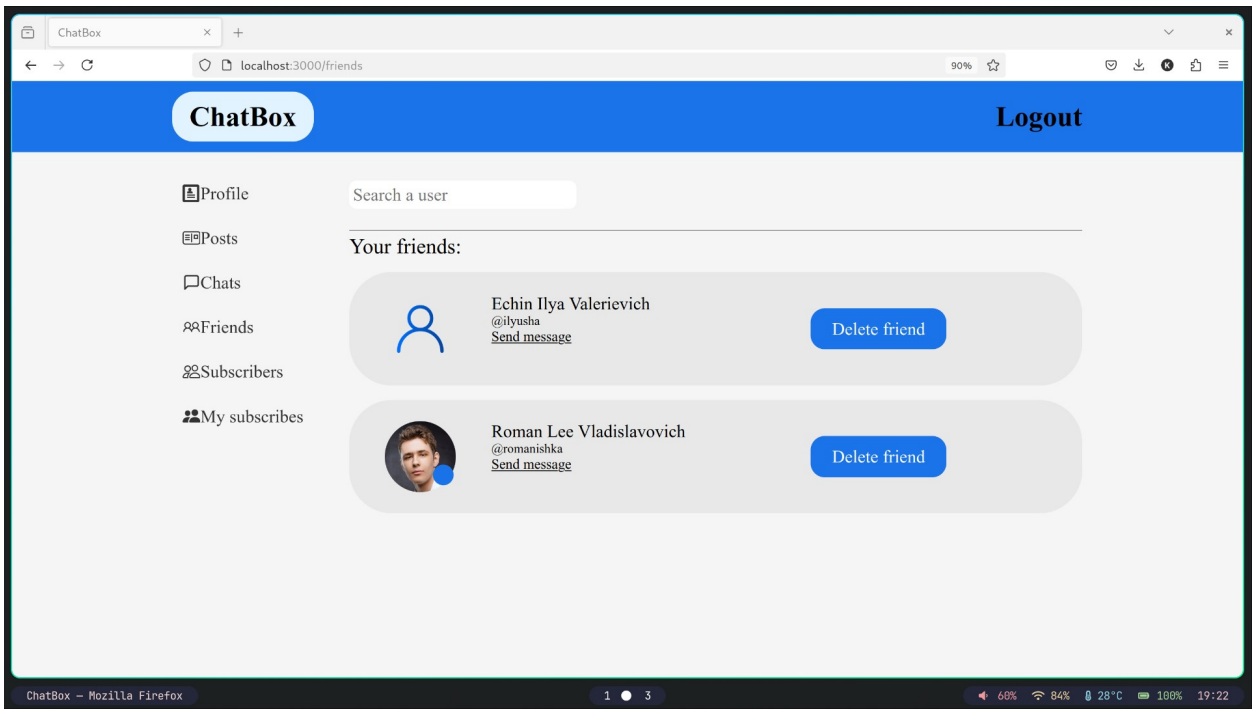

Рис. 9: Друзья

Страницы подписчиков и подписок выглядят аналогично, разное только содержание. На этих страницах можно выполнить поиск по ФИО или нику:

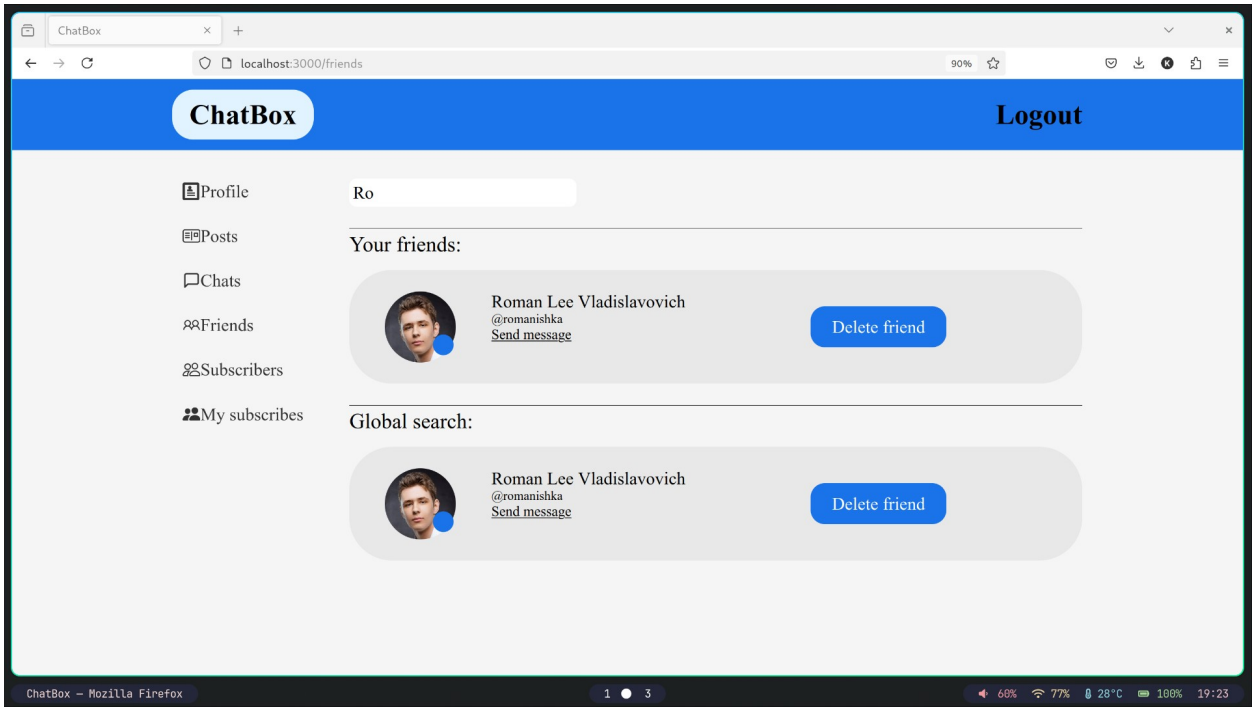

Рис. 10: Поиск по ФИО

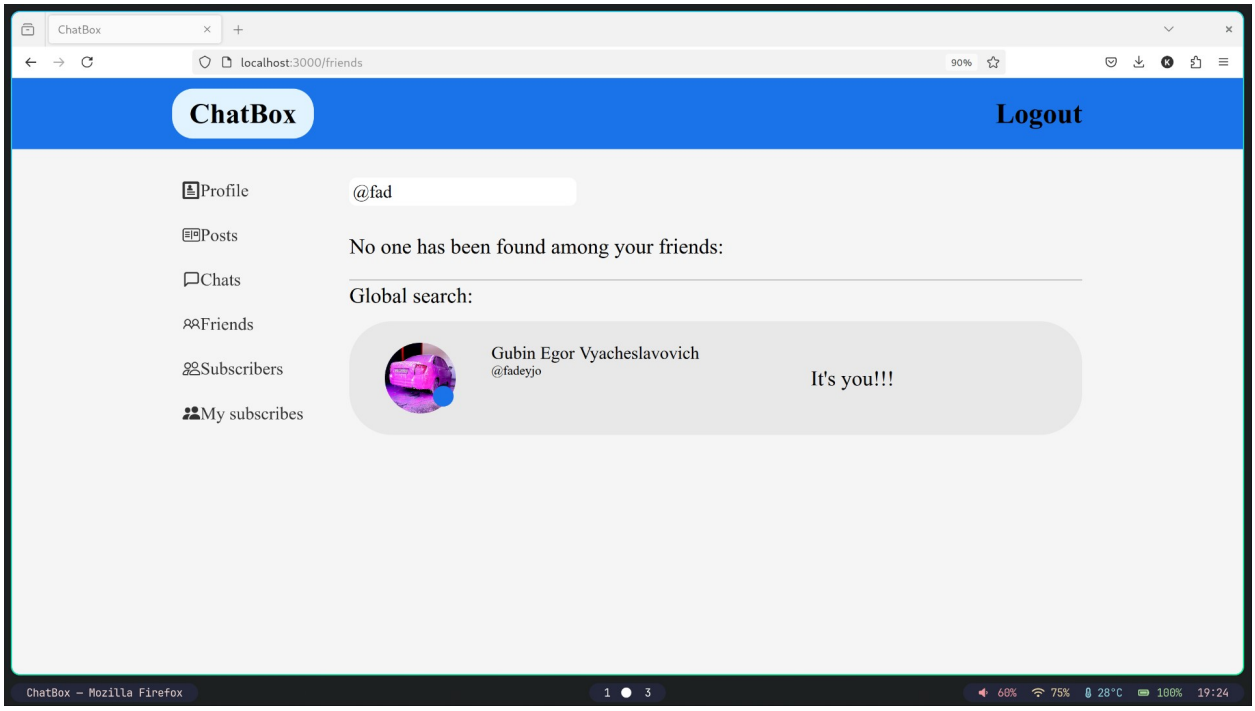

Рис. 11: Поиск по нику

Страница чата:

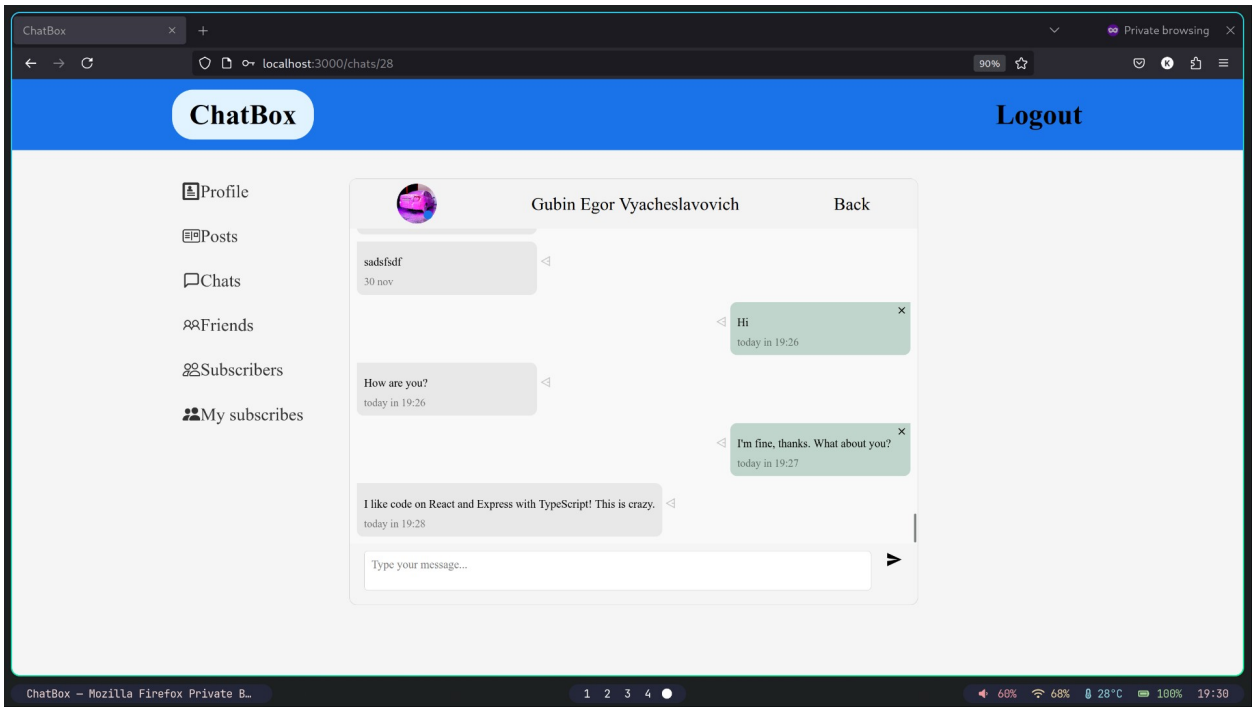

Рис. 12: Страница чата

В любой момент времени от какого-нибудь пользователя может прийти сообщение. Оно отобразится в качестве уведомления:

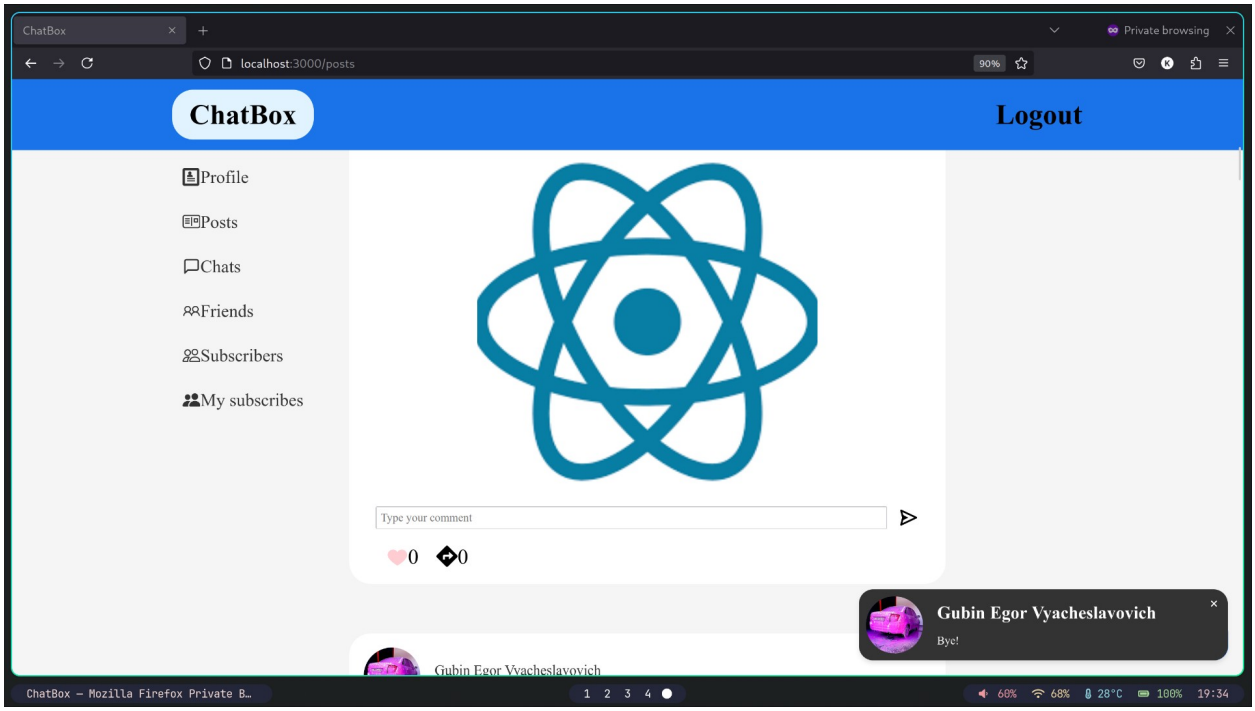

Рис. 13: Уведомление о сообщении

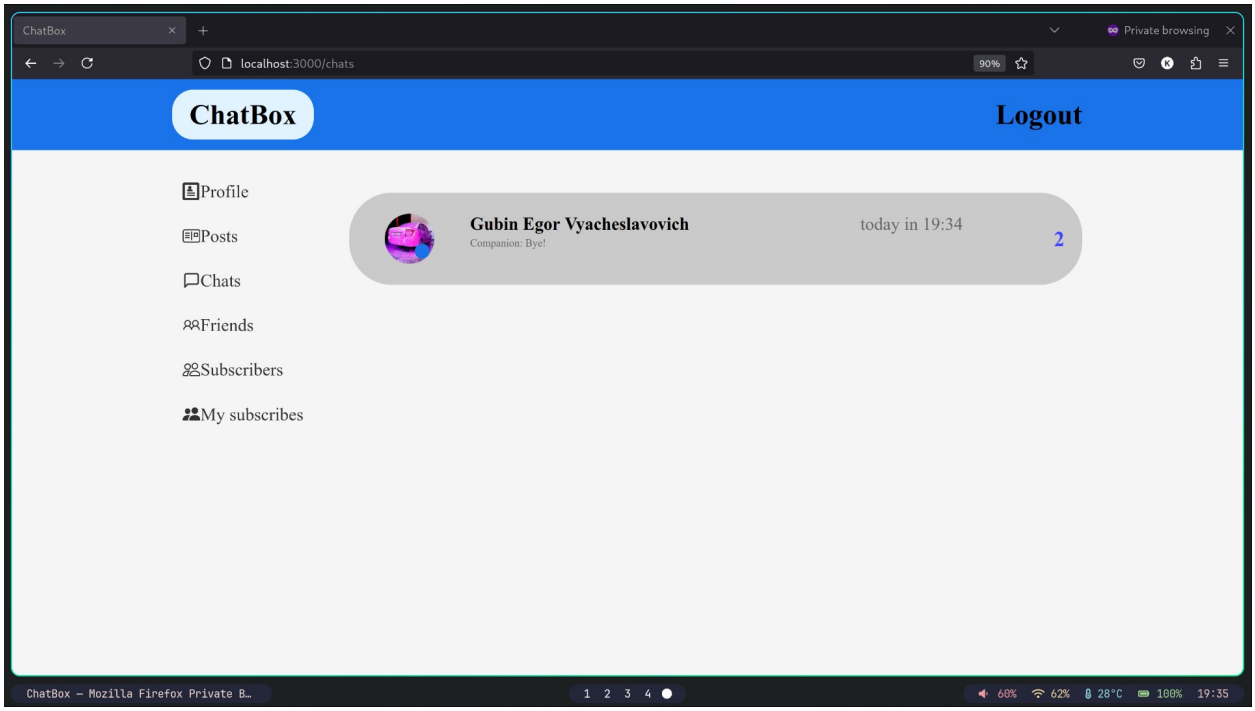

Рис. 14: Авторизованный пользователь не прочитал сообщение

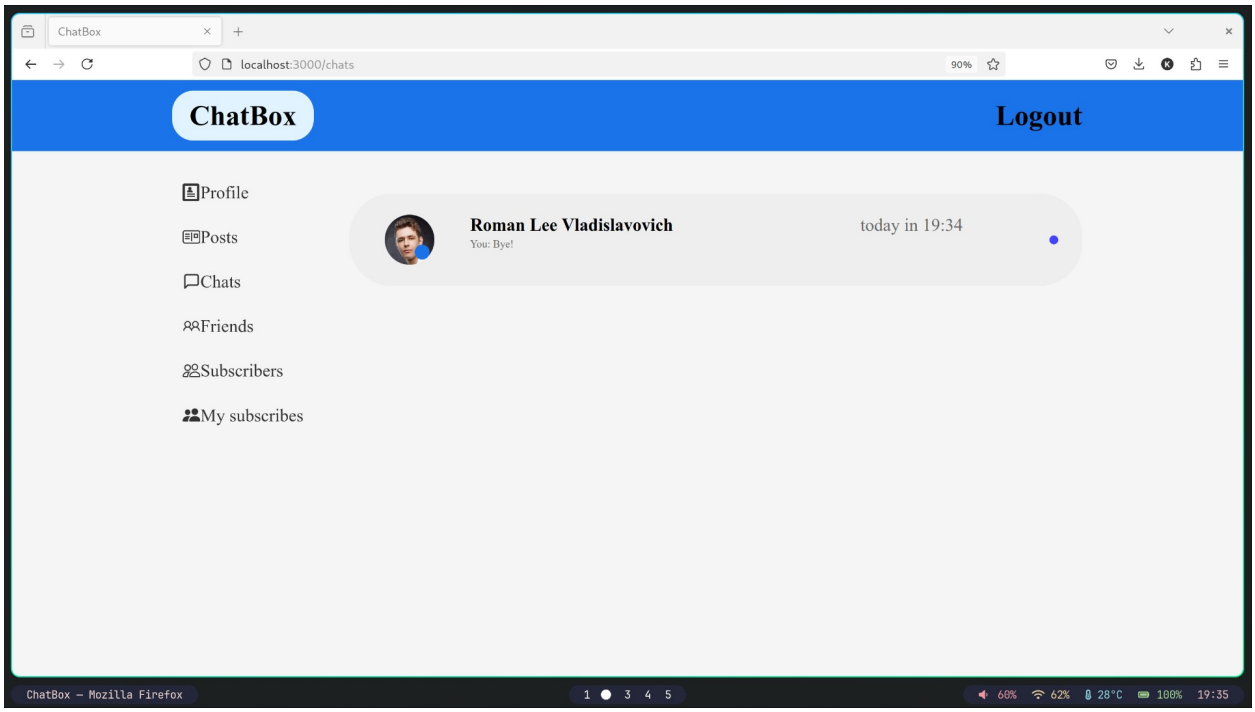

Рис. 15: Собеседник не прочитал сообщение (индикатор в виде точки)

Выводы

В результате было выполнено тестирование компонентов программного продукта. Разработана техническая документация.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была создана социальная сеть, предназначенная для эффективной коммуникации и ведения личного блога. Разработанная система полностью соответствует требованиям технического задания, обеспечивая пользователю удобный и интуитивно понятный интерфейс для общения, публикации и обмена контентом. Система обладает гибкостью и многофункциональностью, позволяя адаптировать платформу под различные нужды пользователей. Внедрение данной социальной сети позволит улучшить взаимодействие между пользователями, повысить вовлеченность аудитории и обеспечить более эффективное ведение личных блогов. Она также способствует экономии времени за счет удобных инструментов для публикации материалов и общения, минимизируя затраты на поддержку традиционных методов взаимодействия.

Поскольку задача была выполнена в полном объеме, дальнейшее расширение системы на данный момент не планируется. Тем не менее, архитектура платформы предусматривает возможность улучшений и расширений, что позволяет адаптировать ее под новые требования в будущем.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Основная литература

1. Романенко, В.В. Объектно-ориентированное программирование : учебное пособие / В.В. Романенко ; Министерство образования и науки Российской Федерации, Томский Государственный Университет Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Томский государственный университет систем управления и радиоэлектроники, 2014. - 475 с. : ил. - Библиогр.: с. 442. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=480517.
2. Николаев, Е.И. Объектно-ориентированное программирование : учебное пособие / Е.И. Николаев ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Северо-Кавказский федеральный университет». - Ставрополь : СКФУ, 2015. - 225 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=458133.
3. Суханов, М.В. Основы Microsoft .NET Framework и языка программирования C# : учебное пособие / М.В. Суханов, И.В. Бачурин, И.С. Майоров ; Министерство образования и науки Российской Федерации, Федеральное государственное автономное образовательное учреждение высшего профессионального образования Северный (Арктический) федеральный университет им. М.В. Ломоносова. - Архангельск : ИД САФУ, 2014. - 97 с. : схем., табл., ил. - Библиогр. в кн. - ISBN 978-5-261-00934-4 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=312313.

Дополнительная литература

1. React.js Documentation: React. *Документация React*. Министерство образования и науки Российской Федерации. - 2024. - URL: https://reactjs.org/docs/getting-started.html.
2. TypeScript Handbook: TypeScript. *The TypeScript Handbook*. Федеральное государственное автономное образовательное учреждение высшего профессионального образования. - 2024. - URL: https://www.typescriptlang.org/docs/.
3. PostgreSQL Documentation: PostgreSQL Global Development Group. *PostgreSQL Documentation*. Министерство образования и науки Российской Федерации. - 2024. - URL: https://www.postgresql.org/docs/.
4. Express.js Documentation: Express. *Express Documentation*. Федеральное государственное автономное образовательное учреждение высшего профессионального образования. - 2024. - URL: https://expressjs.com/en/starter/installing.html.
5. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems: Martin Kleppmann. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media. - 2017. - ISBN 978-1-4919-2862-1.
6. Pro Express.js: Mastering the Express Web Application Framework: Azat Mardan. *Pro Express.js: Mastering the Express Web Application Framework*. Apress. - 2014. - ISBN 978-1-4302-6735-2.
7. Full-Stack React, TypeScript, and Node: David Choi. *Full-Stack React, TypeScript, and Node*. Apress. - 2020. - ISBN 978-1-4842-6341-9.
8. PostgreSQL: Up and Running: Regina Obe, Leo Hsu. *PostgreSQL: Up and Running*. O'Reilly Media. - 2014. - ISBN 978-1-4493-9680-3.

9. React + TypeScript: Best Practices and Tips: Maximilian Schwarzmüller. *React + TypeScript: Best Practices and Tips*. O'Reilly Media. - 2020. - ISBN 978-1-4920-5231-2.
10. Understanding REST APIs with Express and PostgreSQL: The Pragmatic Bookshelf. *Understanding REST APIs with Express and PostgreSQL*. 2020. - URL: https://pragprog.com/titles/dmrest/.