**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

|  |  |
| --- | --- |
| КП.09.02.07-3.25.222.13 ПЗ |  |

**ВЕБ-ПРИЛОЖЕНИЕ «Техническая поддержка»**

|  |  |  |
| --- | --- | --- |
| Председатель ВЦК: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.С. Александрова) |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (Н.Р. Карпова) |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись, дата) | (А.Р. Коротчук) |

Иркутск 2025

Содержание

[Введение 3](#_Toc201136620)

[1. Предпроектное исследование 4](#_Toc201136621)

[1.1 Описание предметной области. 4](#_Toc201136622)

[1.2 Обзор инструментальных средств для разработки 5](#_Toc201136623)

[2 Техническое задание 6](#_Toc201136624)

[3 Проектирование веб-приложения 7](#_Toc201136625)

[3.1 Архитектура программного обеспечения 7](#_Toc201136626)

[3.2 Функциональное и структурное проектирование 10](#_Toc201136627)

[3.3 Проектирование базы данных 17](#_Toc201136628)

[3.4 Проектирование пользовательского интерфейса 21](#_Toc201136629)

[3.4.1 Разработка прототипов интерфейса 21](#_Toc201136630)

[3.4.2 Выбор цветовой гаммы и шрифтов 32](#_Toc201136631)

[3.4.3 Разработка элементов интерфейса 33](#_Toc201136632)

[3.4.4 Разработка дизайна макетов 34](#_Toc201136633)

[4 Разработка веб-приложения 39](#_Toc201136634)

[4.1 Разработка интерфейса веб-приложения 39](#_Toc201136635)

[4.2 Разработка базы данных веб-приложения 42](#_Toc201136636)

[4.3 Разработка серверной части веб-приложения 47](#_Toc201136637)

[5 Документирования программного продукта 53](#_Toc201136638)

[5.1 Руководство пользователя веб-приложения для пользователей 53](#_Toc201136639)

[Заключение 57](#_Toc201136640)

[Список используемых источников 58](#_Toc201136641)

[Приложение А 59](#_Toc201136642)

[Приложение B 65](#_Toc201136643)

# Введение

Хорошая техническая поддержка очень важна в современном мире, обработка обращений клиентов играет большую роль в любой большой компании. Эффективная структура этих процессов позволяет не только поддерживать высокий уровень лояльности клиентов, но и способствует повышению конкурентоспособности предприятия на рынке. Современные технологии предоставляют широкий спектр инструментов для автоматизации и оптимизации работы служб технической поддержки, что делает их использование необходимым условием для обеспечения качественного обслуживания клиентов.

Целью данного курсового проекта является разработка технической поддержки, которая будет эффективно обрабатывать обращения её клиентов. Для достижения этой цели необходимо решить следующие задачи:

1.Провести анализ предметной области.

2. Провести обзор инструментальных средств для разработки.

3.Составить технический анализ

4.Спроектировать базу данных.

5.Разработать прототип, макет для технической поддержки.

6.Разработать веб-приложение Технической поддержки.

Реализация данных задач позволит: создать эффективную техническую поддержку с высоким качеством предоставляемых услуг, повысить лояльность клиентов и увеличить прибыль компании за счет повышения эффективности работы службы технической поддержки.

# 1. Предпроектное исследование

# 1.1 Описание предметной области.

Приложение предназначено для организации взаимодействия между пользователями и службой технической поддержки компании. Оно позволяет пользователям отправлять запросы о проблемах, связанных с продуктами или услугами компании, а сотрудники службы поддержки получают возможность оперативно обрабатывать эти запросы (обращения).

Основные аспекты и компоненты предметной области, которые предназначены для технической поддержки. Роли:

Клиенты: пользователи, обращающиеся за помощью к службе техподдержки.

Атрибуты: имя, фамилия, пароль, контактная информация (электронная почта)

Сотрудники технической поддержки: специалисты, отвечающие за помощь клиентам в решении технических проблем.

Атрибуты: имя, фамилия, пароль, контактная информация (электронная почта), уровень доступа.

Администратор: руководство компании, которое контролирует работу системы.

Атрибуты: имя, пароль, контактная информация (электронная почта), уровень доступа.

Клиенты: Регистрация и авторизация. Создание новых запросов. Просмотр статуса своих обращений. Добавление сообщений в чат.

Сотрудники технической поддержки: Авторизация. Просмотр списка всех активных обращений. Изменение статуса обращений. Добавление сообщений в чат. Закрытие запросов после их решения.

Администраторы: Управление пользователями (изменение прав доступа).

Разработка веб-приложения технической поддержки является важным шагом для повышения качества обслуживания клиентов и оптимизации работы сотрудников службы поддержки. Предложенная архитектура системы включает в себя четкое разделение ролей и полномочий между клиентами, Сотрудниками и администраторами.

# 1.2 Обзор инструментальных средств для разработки

Для разработки сайта технической поддержки, обработки обращений можно использовать различные инструменты и технологии. Вот обзор некоторых популярных инструментов:

1.Фреймворк для бэкенда

Laravel (PHP): Один из самых популярных PHP-фреймворков, предоставляющий удобные средства для работы с базой данных, маршрутизацией и шаблонизацией.

2.Базы данных

MySQL: Популярная база данных, широко используемая благодаря своей совместимости с различными языками программирования.

3.Инструменты для фронтенда

HTML: **это язык гипертекстовой разметки**.  Это стандартный язык разметки, который используется для создания веб-страниц и других типов цифрового контента, который может отображаться в интернете.

CSS: **это язык стилей, с помощью которого оформляют веб-страницы**. Он определяет, как именно должны отображаться элементы HTML, включая макет, цвета, шрифты и другие визуальные аспекты.

VS code: **текстовый редактор для разработчиков, разработанный Microsoft.** Инструмент поддерживает множество языков программирования от JavaScript до Python и предлагает широкий функционал для написания, отладки и тестирования кода.

4.Инструменты для проектирования:

Figma: это **онлайн-сервис для дизайнеров, веб-разработчиков и маркетологов**. Он предназначен для создания прототипов сайтов или приложений, иллюстраций и векторной графики.

Draw.io: это**бесплатный онлайн-сервис**, который позволяет создавать и совместно редактировать диаграммы, различные схемы и прочие визуальные представления данных.

# 2 Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования. Для создания технического задания использовался стандарт ГОСТ 34.602-2020. Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. Введение.
   1. Общие сведения.
   2. Цели и задачи.
2. Основания для разработки.

2.1. Нормативные документы.

2.2. Проектные документы.

1. Назначение системы.
   1. Общее описание.
   2. Преимущества и новизна.
2. Требования к системе.
   1. Функциональные требования.
   2. Технические требования.
   3. Эксплуатационные требования.
3. Требования к техническому обеспечению.
   1. Оборудование.
   2. Сетевые требования.
4. Требования к программному обеспечению.
   1. Программные компоненты.
   2. Интерфейсы.
5. Организационно-технические требования.
   1. Этапы разработки.

Техническое задание на разработку технической поддержки представлено в приложении А.

# 3 Проектирование веб-приложения

# 3.1 Архитектура программного обеспечения

**Архитектура веб-приложения технической поддержки соответствует архитектуре Клиент-сервер:**

**Клиент-серверная архитектура** — это модель организации вычислительных систем, в которой задачи распределены между клиентами и серверами. В такой архитектуре клиент, обычно являющийся пользователем или программой, запрашивает услуги или ресурсы у сервера, который отвечает на запросы, предоставляя необходимые данные или функциональность.

Так как в работе будет использован Laravel, то важно пояснить, что такое архитектурный паттерн MVC:

MVC (Model-View-Controller) – это архитектурный паттерн, который используется для разделения приложения на три взаимосвязанные части: модель, представление и контроллер. Цель этого разделения – упрощение разработки и поддержание кода, а также улучшение его тестируемости и расширяемости.  
Модель (Model)

Модель представляет собой слой данных приложения. Она содержит логику работы с данными, включая их хранение, получение, изменение и удаление. В модели находятся сущности, которые представляют реальные объекты предметной области (например, пользователь, заказ, товар и т.п.). Модель не зависит от представления и контроллера; она занимается только управлением данными и бизнес-логикой.

Представление (View)

Представление отвечает за отображение данных пользователю. Оно формирует интерфейс, с которым взаимодействует пользователь, будь то веб-страница, мобильное приложение или другое визуальное представление. Представление получает данные от контроллера и преобразует их в удобный для восприятия вид.  
Контроллер (Controller)

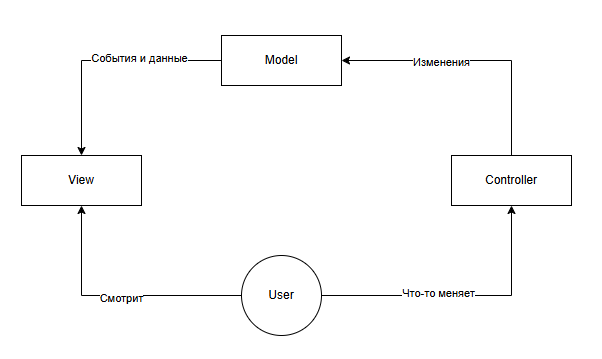


Рисунок 1 – Модель MVC

Контроллер является связующим звеном между моделью и представлением. Он обрабатывает входные данные от пользователя, передает их модели для обработки, а затем обновляет представление на основе результатов обработки.

Проектирование архитектуры (построить клиент-серверную архитектуру)

Архитектура будет состоять из трех основных уровней:

1.Интерфейсный уровень (Frontend):

2.Отображение данных пользователю.

3.Обработка действий пользователя.

Уровень логики приложения (Backend):

1.Обработка запросов пользователя.

2.Управление данными.

3.Аутентификация и авторизация пользователей.

Уровень данных (Database):

Хранение информации о пользователях, обращениях, ответах.

1.Система управления обращениями

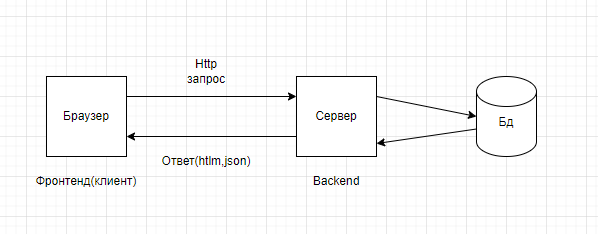


Рисунок 2 – клиент-серверная архитектура

Основная часть архитектуры, которая отвечает за обработку запросов пользователей. Она включает следующие функции:

1.Регистрация ;

2. Передача обращений сотрудникам;

3.Отслеживание статуса обращений;

4.Коммуникация с пользователем по поводу решения проблемы.

2.База данных

Хранение информации о пользователях, обращениях и статусе их выполнения и т.п. Базы данных должны обеспечивать высокую производительность и надежность хранения данных.

3.Пользовательский интерфейс

Это веб-интерфейс через которое пользователи могут отправлять запросы на техническую поддержку, оформлять обращения.

# 3.2 Функциональное и структурное проектирование

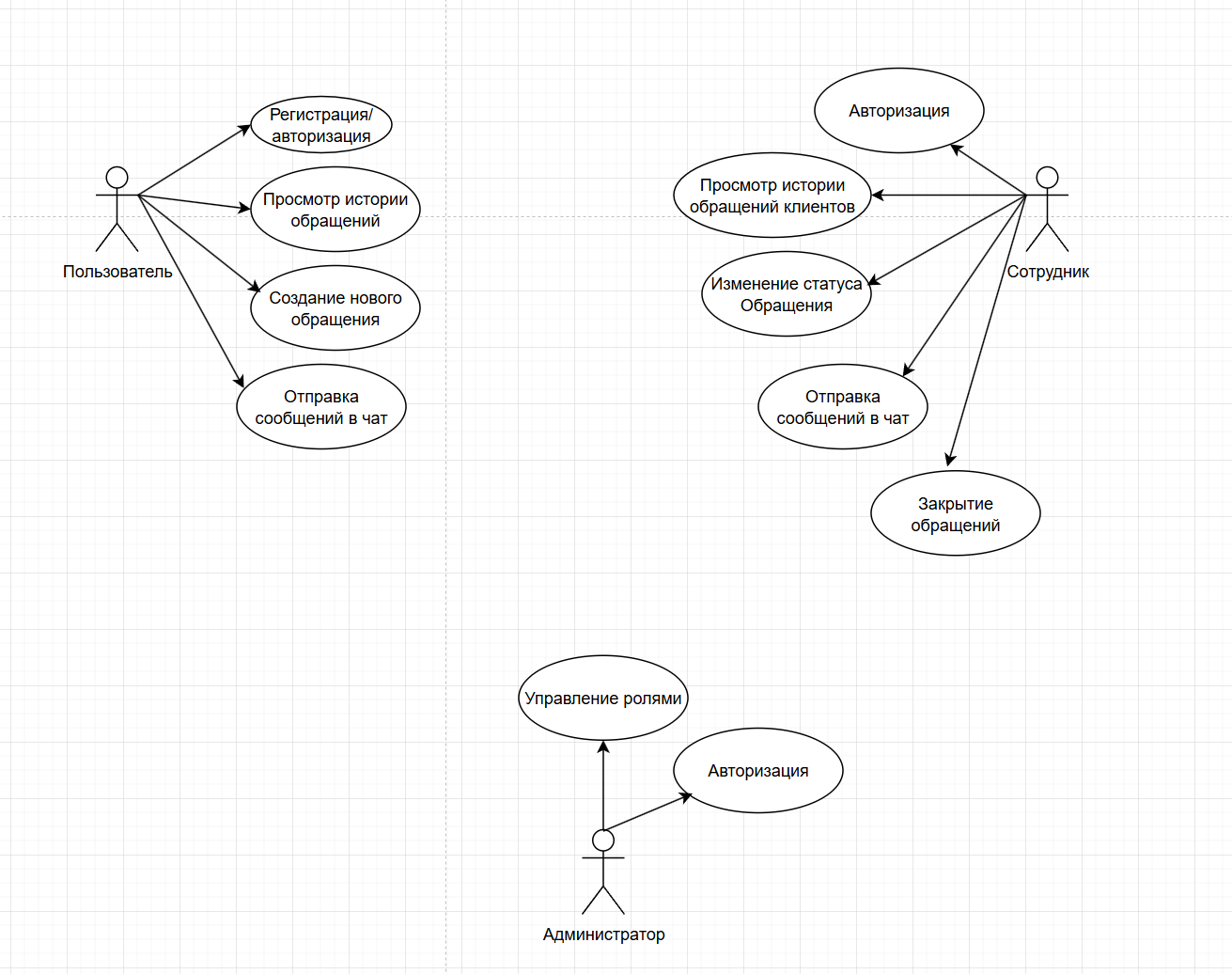


Рисунок 3 – Диаграмма прецедентов

Диаграмма прецедентов демонстрирует взаимодействие акторов с системой и основные функциональные возможности.

Actors:

Клиент : Пользователь, обращающийся за помощью.

Сотрудник технической поддержки : Специалист, решающий проблемы клиентов.

Администратор : Руководитель, контролирующий работу системы.

Основные прецеденты: Клиент : Регистрация/Авторизация. Создание нового обращения. Просмотр статуса обращения. Добавление сообщений в чат.

Сотрудник технической поддержки : Авторизация. Просмотр истории обращений клиентов. Изменение статуса обращения. Добавление сообщений в чат. Закрытие обращений.

Администратор : Управление ролями.

Описание диаграммы: На диаграмме отображаются связи между акторами и прецедентами, а также зависимости между прецедентами (например, "Регистрация" является необходимым шагом перед "Созданием обращения").

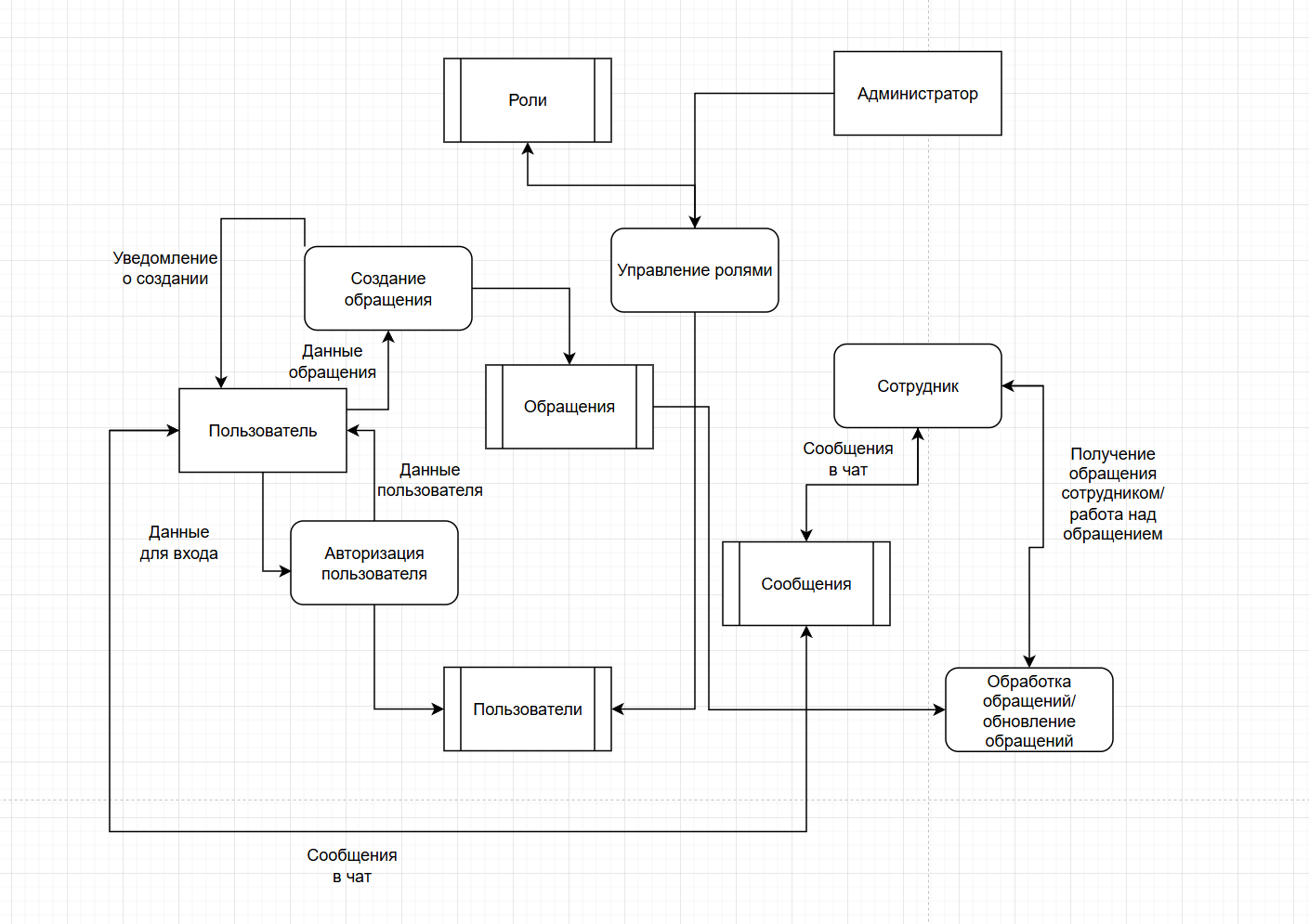


Рисунок 4 – Диаграмма потока данных

Диаграмма потока данных показывает, как данные перемещаются между внешними сущностями, процессами и базами данных.

Уровень 0 (Общий вид): Внешние сущности : Клиент. Сотрудник технической поддержки. Администратор.

Процессы : Обработка запросов. Управление ролями. Базы данных : База Пользователей. База запросов.

Уровень 1 (Детальный анализ): Поток данных для клиента : Создание обращения → База обращений. Получение статуса обращения ← База обращений.

Поток данных для сотрудника технической поддержки : Обработка/обновление обращений ← База обращений. Обновление статуса обращений → База обращений.

Поток данных для администратора : Управление пользователями → База пользователей.

Описание диаграммы: Диаграмма наглядно демонстрирует движение данных через систему, помогая выявить ключевые точки обработки информации.

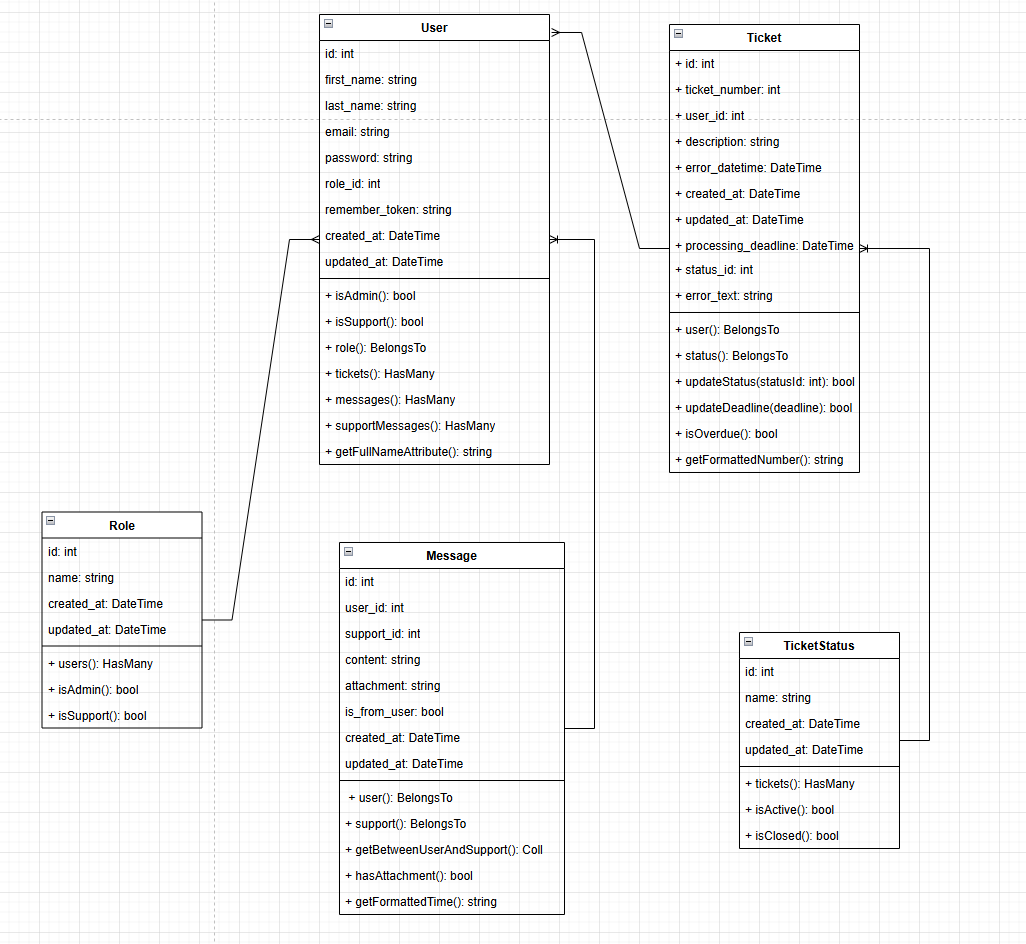


Рисунок 5 – Диаграмма классов

Диаграмма классов описывает структуру системы, показывая взаимосвязи между классами.

User (Пользователь):

Атрибуты: id, first\_name, last\_name, email, password, role\_id, remember\_token, created\_at, updated\_at.

Методы: register(), login(), createTicket(), sendMessage(), isAdmin(), isSupport(), getFullName().

Role (Роль):

Атрибуты: id, name, created\_at, updated\_at.

Методы: assignToUser(), getUsersByRole(), isAdmin(), isSupport().

Ticket (Обращение):

Атрибуты: id, ticket\_number, user\_id, description, error\_text, error\_datetime, processing\_deadline, status\_id, created\_at, updated\_at.

Методы: create(), updateStatus(), updateDeadline(), assignToSupport(), isOverdue(), getFormattedNumber().

TicketStatus (Статус обращения):

Атрибуты: id, name, created\_at, updated\_at.

Методы: updateTicketStatus(), getTicketsByStatus(), isActive(), isClosed().

Message (Сообщение):

Атрибуты: id, user\_id, support\_id, content, attachment, is\_from\_user, created\_at, updated\_at.

Методы: create(), getBetweenUserAndSupport(), markAsRead(), sendToUser(), hasAttachment(). Описание диаграммы: Классы связаны между собой через отношения ассоциации, например: User связан с Role через отношение "многие к одному" (много пользователей могут иметь одну роль). User связан с Message через отношение "один ко многим" как сотрудник поддержки (один сотрудник поддержки может получить много сообщений).

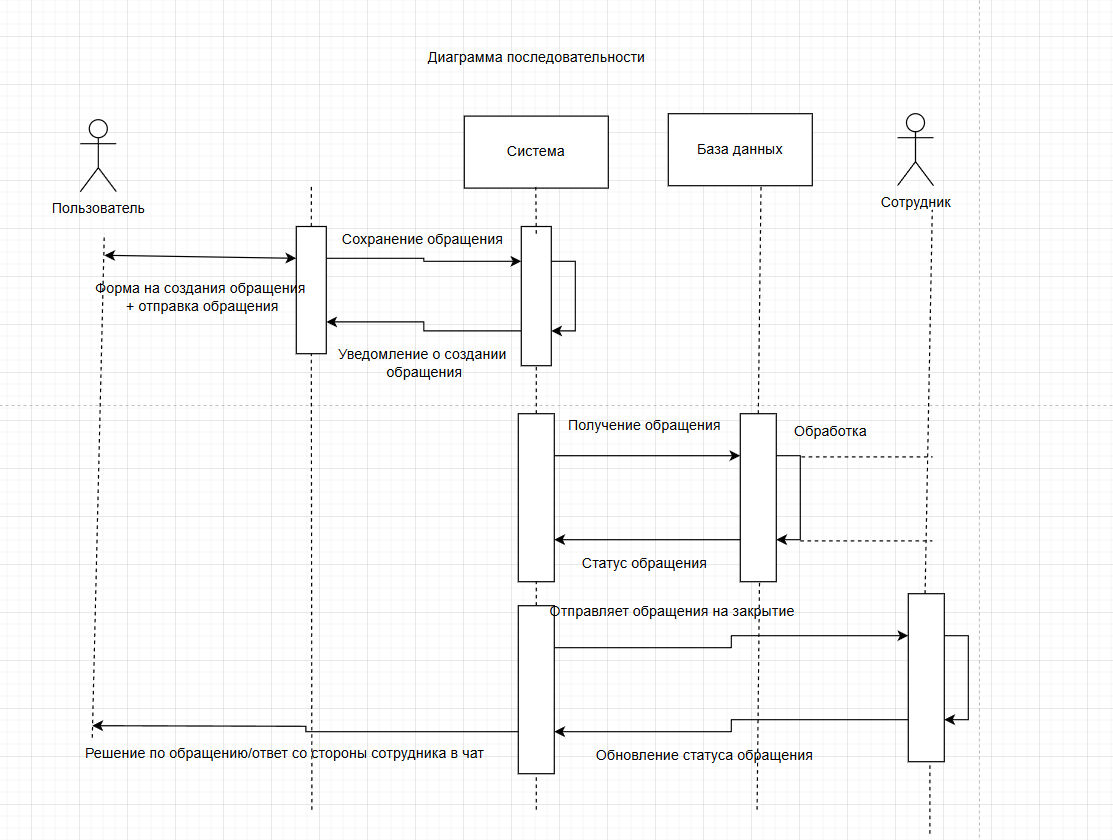


Рисунок 6 – Диаграмма последовательности

Диаграмма последовательности описывает взаимодействие объектов в определенном порядке.

Пример сценария: Создание обращения клиентом Клиент → Система: Отправляет запрос. Система → База данных: Ведёт новый запрос в обработку.

Пример сценария: Закрытие запроса сотрудником Сотрудник → Система: Выбирает запрос для закрытия. Система → База данных: Обновляет статус обращения. Система → Клиент: Отправляет решение по обращению (ответ со стороны сотрудника в чат).

Описание диаграммы: Диаграмма последовательности позволяет детально проанализировать логику работы системы и выявить возможные ошибки или недочеты.

Выводы по построению диаграмм. Диаграмма прецедентов помогает четко определить функциональные требования системы и роли участников. Диаграмма потока данных обеспечивает понимание того, как данные перемещаются внутри системы и взаимодействуют с внешними сущностями. Диаграмма классов дает подробное описание структуры системы и взаимосвязей между компонентами. Диаграмма последовательности позволяет протестировать логику работы системы и найти потенциальные проблемы.

Таким образом, представленные диаграммы обеспечивают комплексный подход к разработке веб-приложения технической поддержки, что способствует повышению качества обслуживания клиентов и эффективности работы службы поддержки.

# **3.3 Проектирование базы данных**

Проектирование базы данных является ключевым этапом разработки веб-приложения для технической поддержки. ER-модель (рисунок 7) используется для визуального представления данных и их структуры посредством описания сущностей, атрибутов и связей между ними.

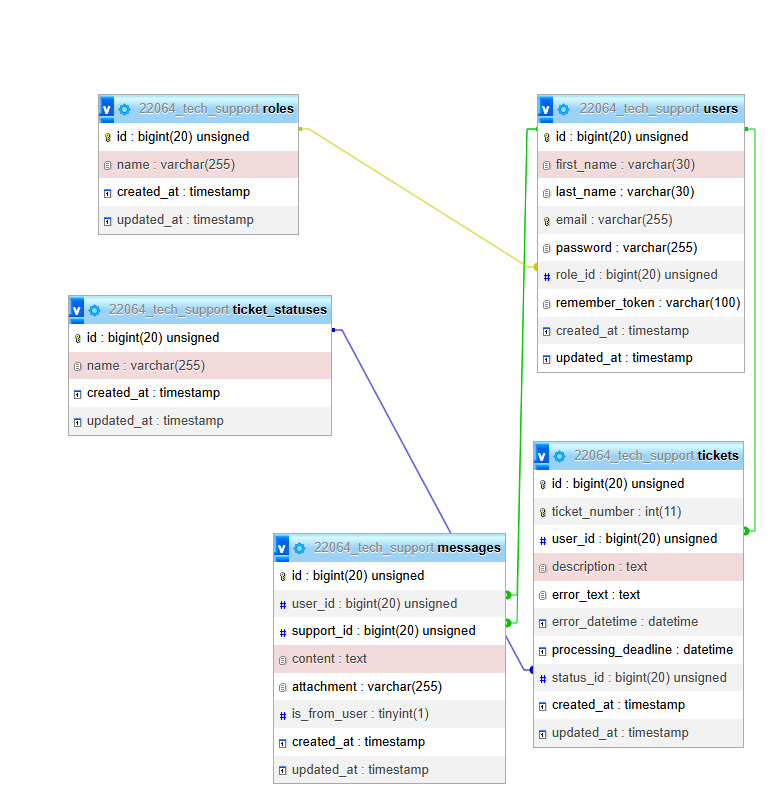


Рисунок 7 – ER-модель

Таблица 1: Users (Пользователи)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ПОЛЕ** | **ТИП ДАННЫХ** | **ОПИСАНИЕ** | **ПЕРВИЧНЫЙ КЛЮЧ** | **ВНЕШНИЙ КЛЮЧ** |
| id | BIGINT(20) UNSIGNED | Уникальный идентификатор пользователя | Да | Нет |
| first\_name | VARCHAR(30) | Имя пользователя | Нет | Нет |
| last\_name | VARCHAR(30) | Фамилия пользователя | Нет | Нет |
| email | VARCHAR(255) | Электронная почта пользователя | Нет | Нет |
| password | VARCHAR(255) | Пароль пользователя (зашифрованный) | Нет | Нет |
| role\_id | BIGINT(20) UNSIGNED | ID роли (ссылка на таблицу Roles) | Нет | Да |
| remember\_token | VARCHAR(100) | Токен для запоминания сессии | Нет | Нет |
| created\_at | TIMESTAMP | Дата и время создания записи | Нет | Нет |
| updated\_at | TIMESTAMP | Дата и время последнего обновления | Нет | Нет |

Таблица 2: Roles (Роли)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ПОЛЕ** | **ТИП ДАННЫХ** | **ОПИСАНИЕ** | **ПЕРВИЧНЫЙ КЛЮЧ** | **ВНЕШНИЙ КЛЮЧ** |
| id | BIGINT(20) UNSIGNED | Уникальный идентификатор роли | Да | Нет |
| name | VARCHAR(255) | Название роли (например, "пользователь", "сотрудник поддержки", "администратор") | Нет | Нет |
| created\_at | TIMESTAMP | Дата и время создания записи | Нет | Нет |
| updated\_at | TIMESTAMP | Дата и время последнего обновления | Нет | Нет |

Таблица 3: Tickets (обращения)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ПОЛЕ** | **ТИП ДАННЫХ** | **ОПИСАНИЕ** | **ПЕРВИЧНЫЙ КЛЮЧ** | **ВНЕШНИЙ КЛЮЧ** |
| id | BIGINT(20) UNSIGNED | Уникальный идентификатор обращения | Да | Нет |
| ticket\_number | INT(11) | Номер обращения для пользователя | Нет | Нет |
| user\_id | BIGINT(20) UNSIGNED | ID пользователя (ссылка на таблицу Users) | Нет | Да |
| description | TEXT | Описание проблемы | Нет | Нет |
| error\_text | TEXT | Текст ошибки | Нет | Нет |
| error\_datetime | DATETIME | Дата и время возникновения ошибки | Нет | Нет |
| processing\_deadline | DATETIME | Крайний срок обработки обращения | Нет | Нет |
| status\_id | BIGINT(20) UNSIGNED | ID статуса (ссылка на таблицу TicketStatuses) | Нет | Да |
| created\_at | TIMESTAMP | Дата и время создания записи | Нет | Нет |
| updated\_at | TIMESTAMP | Дата и время последнего обновления | Нет | Нет |

Таблица 4: Message (Сообщения)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ПОЛЕ** | **ТИП ДАННЫХ** | **ОПИСАНИЕ** | **ПЕРВИЧНЫЙ КЛЮЧ** | **ВНЕШНИЙ КЛЮЧ** |
| id | BIGINT(20) UNSIGNED | Уникальный идентификатор сообщения | Да | Нет |
| user\_id | BIGINT(20) UNSIGNED | ID пользователя (ссылка на таблицу Users) | Нет | Да |
| support\_id | BIGINT(20) UNSIGNED | ID сотрудника поддержки (ссылка на таблицу Users) | Нет | Да |
| content | TEXT | Содержимое сообщения | Нет | Нет |
| attachment | VARCHAR(255) | Путь к прикрепленному файлу | Нет | Нет |
| is\_from\_user | TINYINT(1) | Флаг: сообщение от пользователя (1) или от поддержки (0) | Нет | Нет |
| created\_at | TIMESTAMP | Дата и время создания записи | Нет | Нет |
| updated\_at | TIMESTAMP | Дата и время последнего обновления | Нет | Нет |

Таблица 5: Ticket-Statuses (Статусы обращения)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ПОЛЕ** | **ТИП ДАННЫХ** | **ОПИСАНИЕ** | **ПЕРВИЧНЫЙ КЛЮЧ** | **ВНЕШНИЙ КЛЮЧ** |
| id | BIGINT(20) UNSIGNED | Уникальный идентификатор статуса | Да | Нет |
| name | VARCHAR(255) | Название статуса (например, "новое", "в работе", "закрыто") | Нет | Нет |
| created\_at | TIMESTAMP | Дата и время создания записи | Нет | Нет |
| updated\_at | TIMESTAMP | Дата и время последнего обновления | Нет | Нет |

# 3.4 Проектирование пользовательского интерфейса

# 3.4.1 Разработка прототипов интерфейса

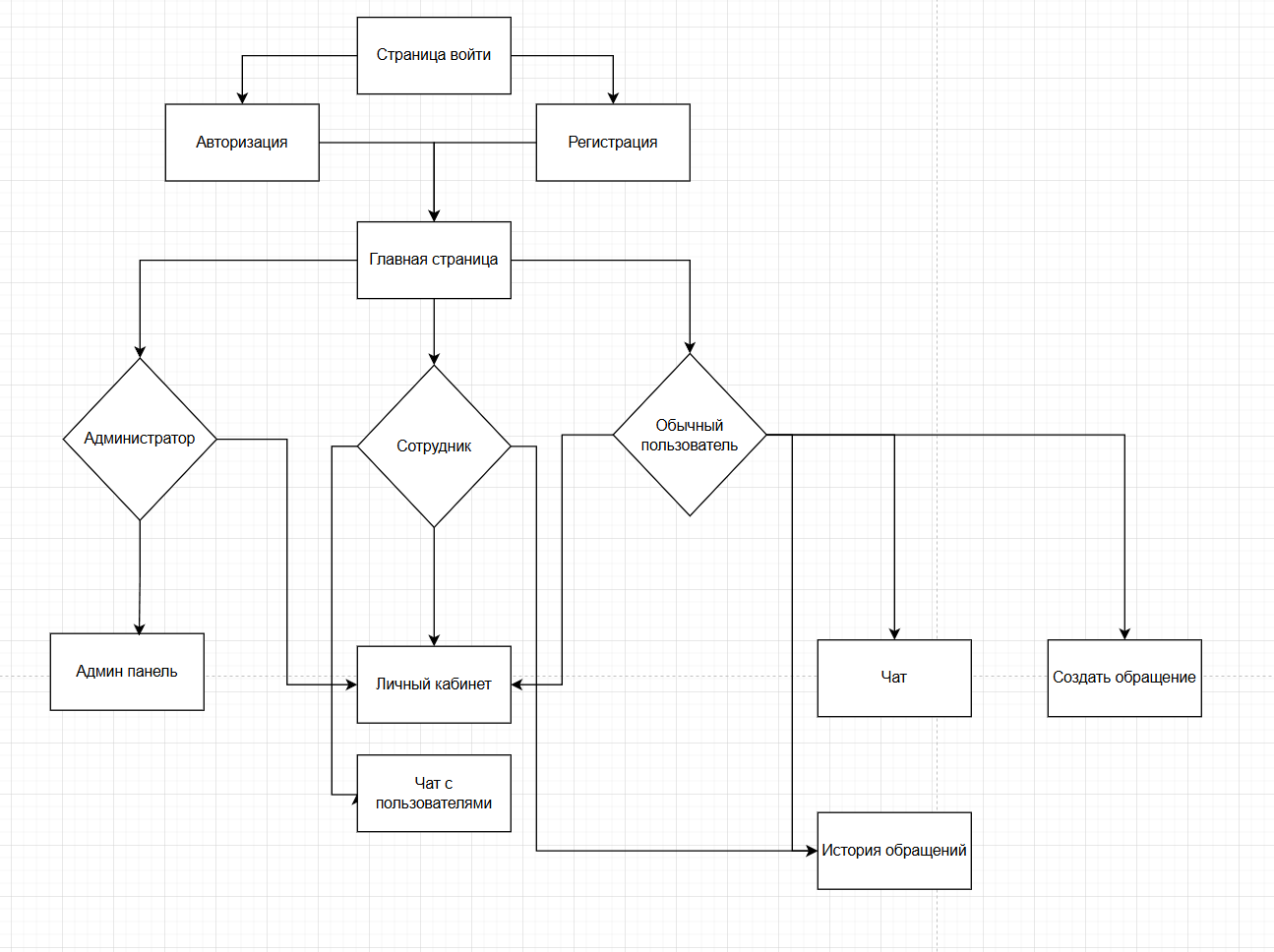
****

Рисунок 8 *–* Схема навигации

Схема навигации показывает основные страницы веб-приложения «Техническая поддержка» и их взаимосвязь, представлена на рисунке 8. Любой гость системы начинает со страницы входа, где он может авторизоваться или зарегистрироваться. Далее уже пользователя переносит на главную страницу, где в зависимости от его роли он может перейти на разные страницы, если это обычный пользователь, то он может создать обращения, посмотреть своё обращение в истории и написать в чат сотруднику. Если пользователь c ролью сотрудник, то он может заходить в чат с пользователем и в историю обращений. Если в систему зашёл администратор, то он имеет доступ в админ панель. Также каждый из пользователей системы с ролью может перейти в личный кабинет.

Прототип играет ключевую роль в создании технической поддержки. Он представляет собой черновую версию сайта, позволяющую наглядно представить и протестировать концепцию, функциональность и взаимодействие с пользователями до начала основного этапа разработки.

На рисунке 9 представлен прототип главной страницы, где сверху написано “техническая поддержка”, справа сверху показана кнопка “личный кабинет”. В центре страницы есть кнопки сверху вниз “Обращения”, “Создать обращения”, “Чат”. Слева сверху и снизу есть логотип сайта.

****

Рисунок 9 – Прототип главной страницы

На рисунке 10 предоставлен прототип чата. На странице изображен чат, где сотрудник и пользователь системы могут общаться друг с другом для решения проблемы. В чат можно отправлять фотографии с помощью специальной кнопки.

****

Рисунок 10 – Прототип чата

На рисунке 11 предоставлен прототип “создания обращения”, где есть текстовые поля, куда нужно вводить нужную информацию “Дата и время возникновения ошибки”, “Описание проблемы”, “Текст ошибки” и кнопка внизу “создать обращение”.

****

Рисунок 11 – Прототип создания обращения

На рисунке 12 предоставлен прототип “история обращения”, после кнопки создать обращение мы переходим на эту страницу и созданное обращение появляется здесь со всей информацией, которая ранее была введена на прошлой странице.

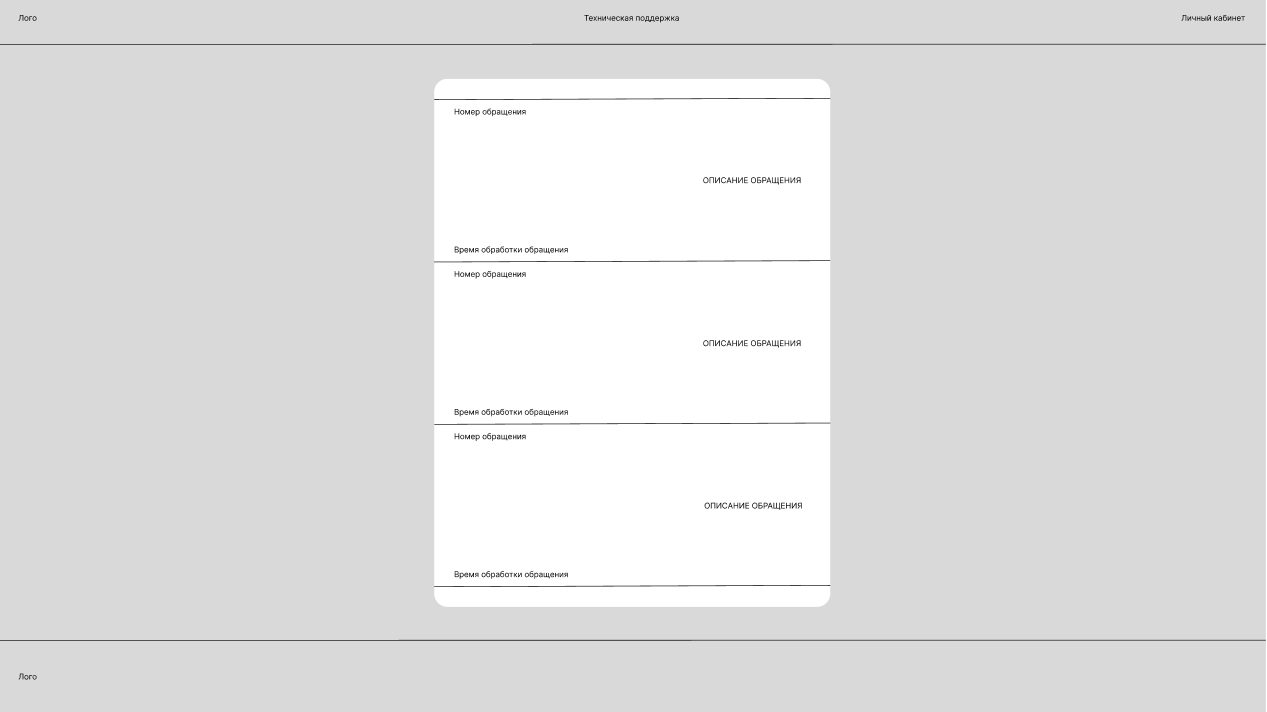
****

Рисунок 12 – Прототип история обращений

На рисунке 13 предоставлен прототип страница “Админка”. На странице есть блок для управления ролями пользователей. Слева в блоке есть информация о пользователе ФИ (фамилия и имя), а справа его роль и подтверждение, чтобы сменить роль и подтвердить это действие.

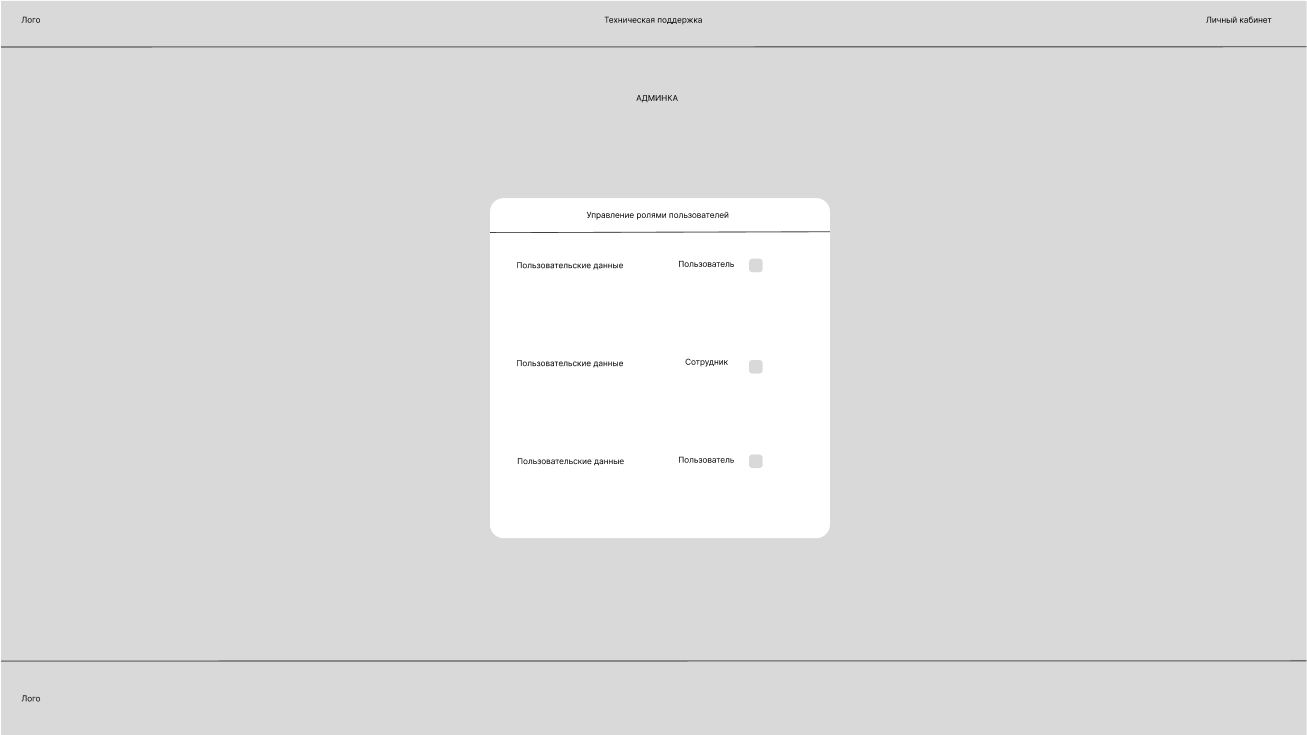
****

Рисунок 13 – Прототип админка

На рисунке 14 предоставлен прототип регистрации. На странице изображён блок с текстовыми полями сверху вниз, где нужно вводить данные чтобы зарегистрироваться: “имя”, ”фамилия”, “email”, “пароль” и кнопки “Зарегистрироваться” и “уже есть аккаунт”

****

Рисунок 14 – Прототип регистрация

На рисунке 15 предоставлен прототип авторизации. На блоке изображены текстовые поля сверху вниз, где нужно заполнить информацию для авторизации в аккаунт: “Email”, “пароль”. Также есть кнопки “Забыл пароль”, “войти”, “создать аккаунт”

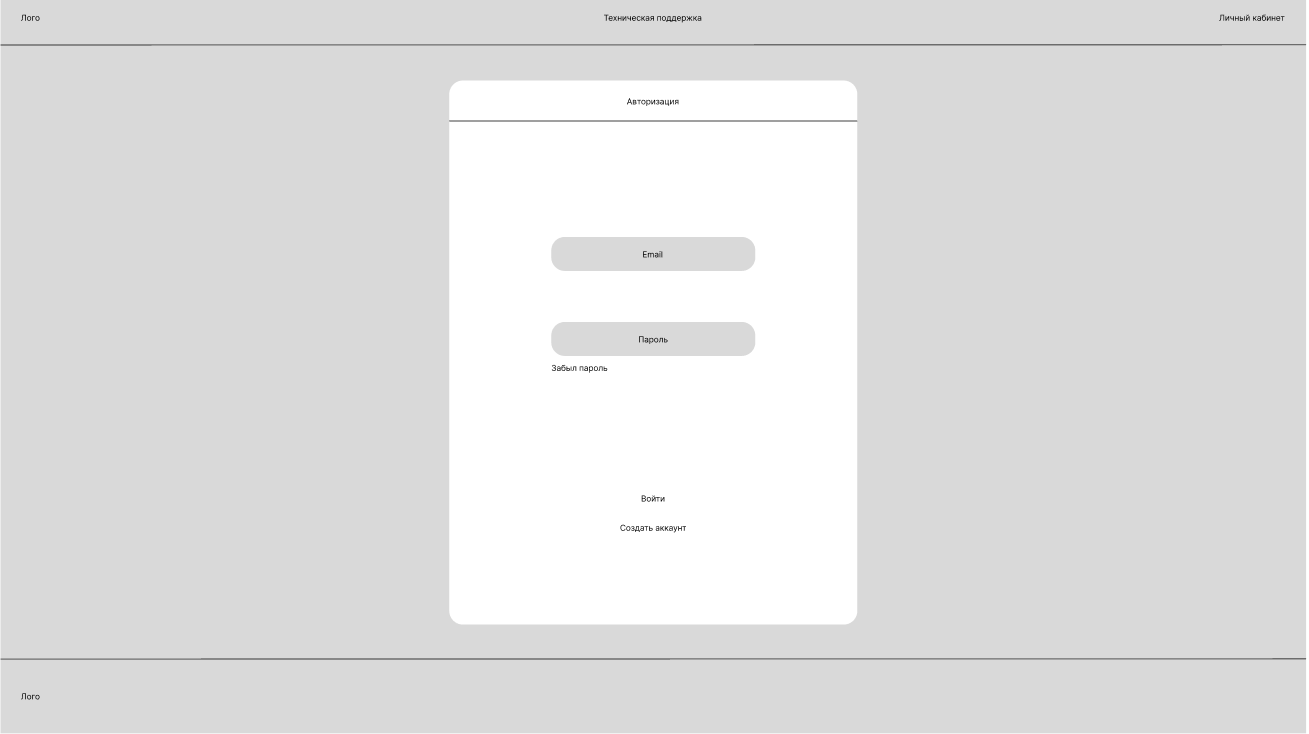


Рисунок 15 – Прототип авторизация

На рисунке 16 предоставлен прототип забыл пароль. На блоке изображены текстовые поля сверху вниз, где нужно заполнить информацию для авторизации в аккаунт: “Email”, “новый пароль”, “подтверждение пароля”. Также есть кнопки “изменить”, “я помню пароль”

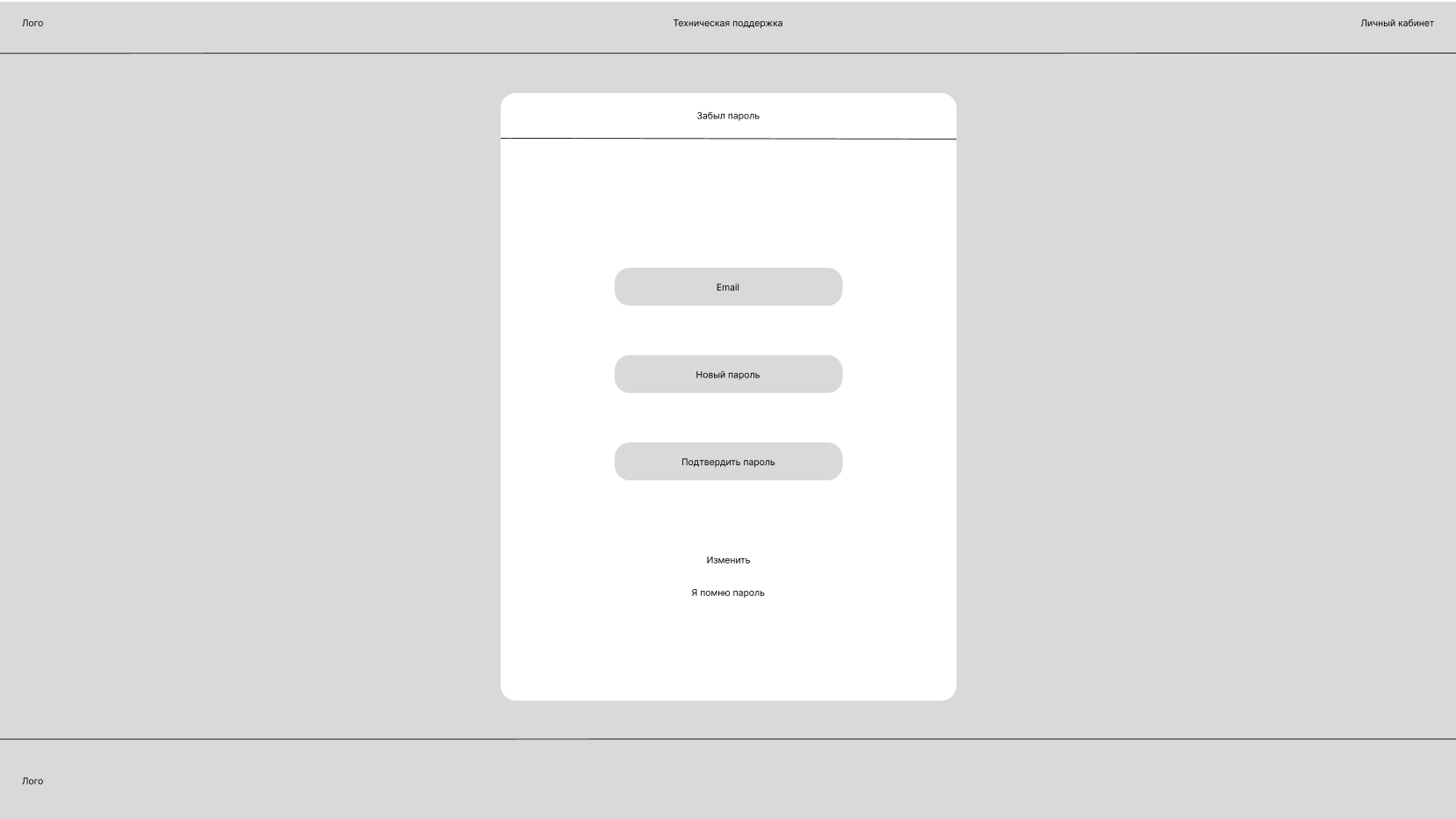


Рисунок 16 – Прототип забыл пароль

На рисунке 17 предоставлен прототип личного кабинета. Страница разделена на два блока слева личные данные пользователя, которые можно изменять. Данные написаны сверху вниз, а рядом кнопки изменить: “имя”, ”фамилия”, “email”, “пароль”. Справа кнопки для перехода на другие страницы. “Обращения”, “Создать обращения”, “Чат”.

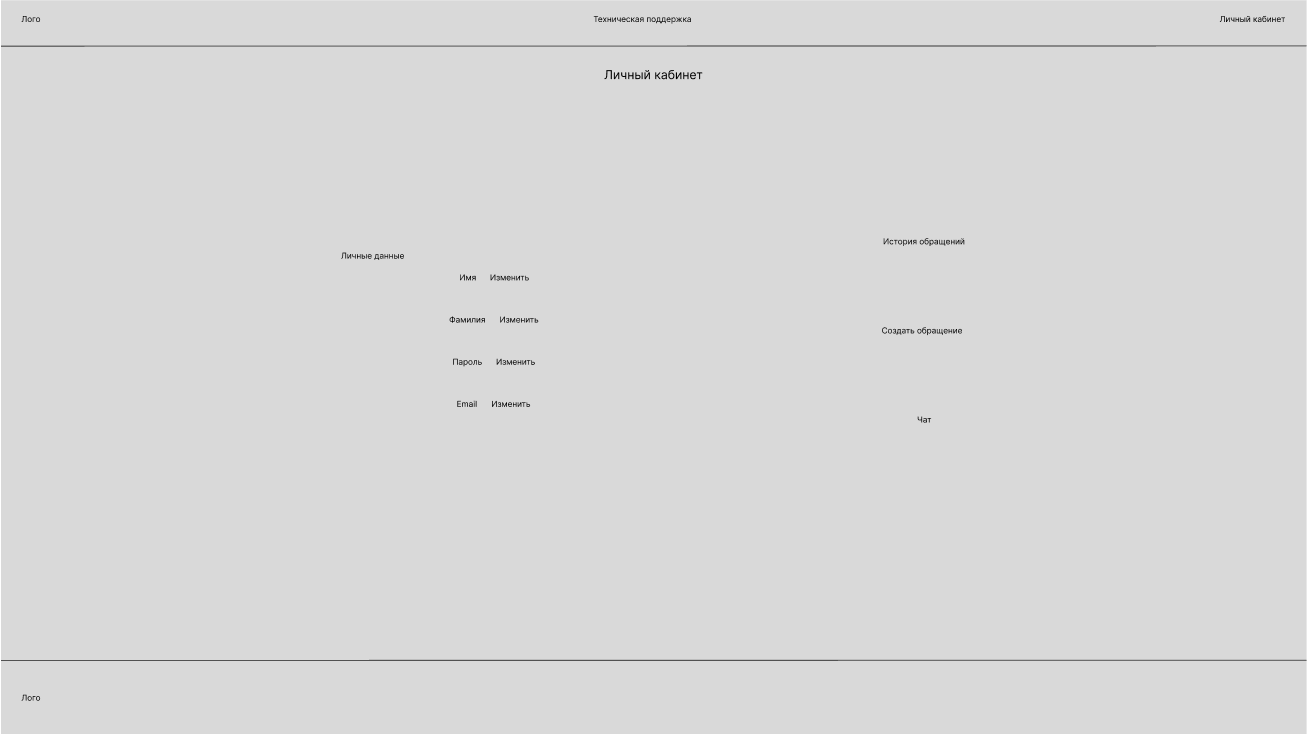


Рисунок 17 – Прототип личный кабинет

На рисунке 18 предоставлена мобильная версия прототипа главной страницы. Из нового кнопка “личный кабинет” заменена на иконку. Также добавлена снизу справа кнопка в виде иконки для возврата на главную страницу (Также добавлено пояснение на главной странице, что делает эта кнопка)



Рисунок 18 – главная страница (мобильная версия)

# 3.4.2 Выбор цветовой гаммы и шрифтов

На рисунке 19 изображена цветовая гамма и выбранный шрифт.

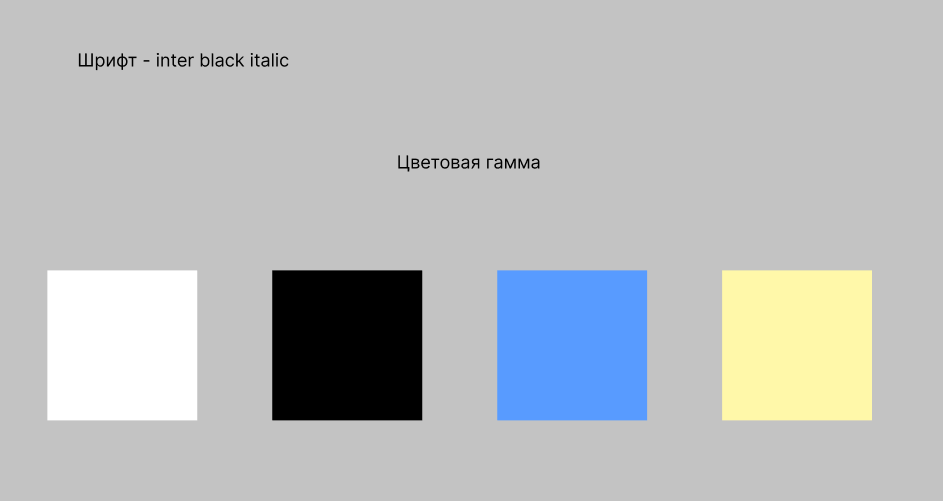
****

Рисунок 19 – Цветовая гамма и шрифт

Цветовая гамма и шрифты были выбраны для минимализма и удобства пользователей. Цвета синий и желтый были выбраны только для страницы чат, чтобы отобразить цвет пользователя в чате (синий) и сотрудника (желтый).

# 3.4.3 Разработка элементов интерфейса



Рисунок 20 – Кнопка (иконка) личного кабинета.



Рисунок 21 – Кнопка (иконка) загрузки файлов в чат.



Рисунок 22 – Кнопка (иконка) для отправки в чат сообщений.



Рисунок 23 – Кнопка (иконка) для возврата на главную страницу.

# 3.4.4 Разработка дизайна макетов

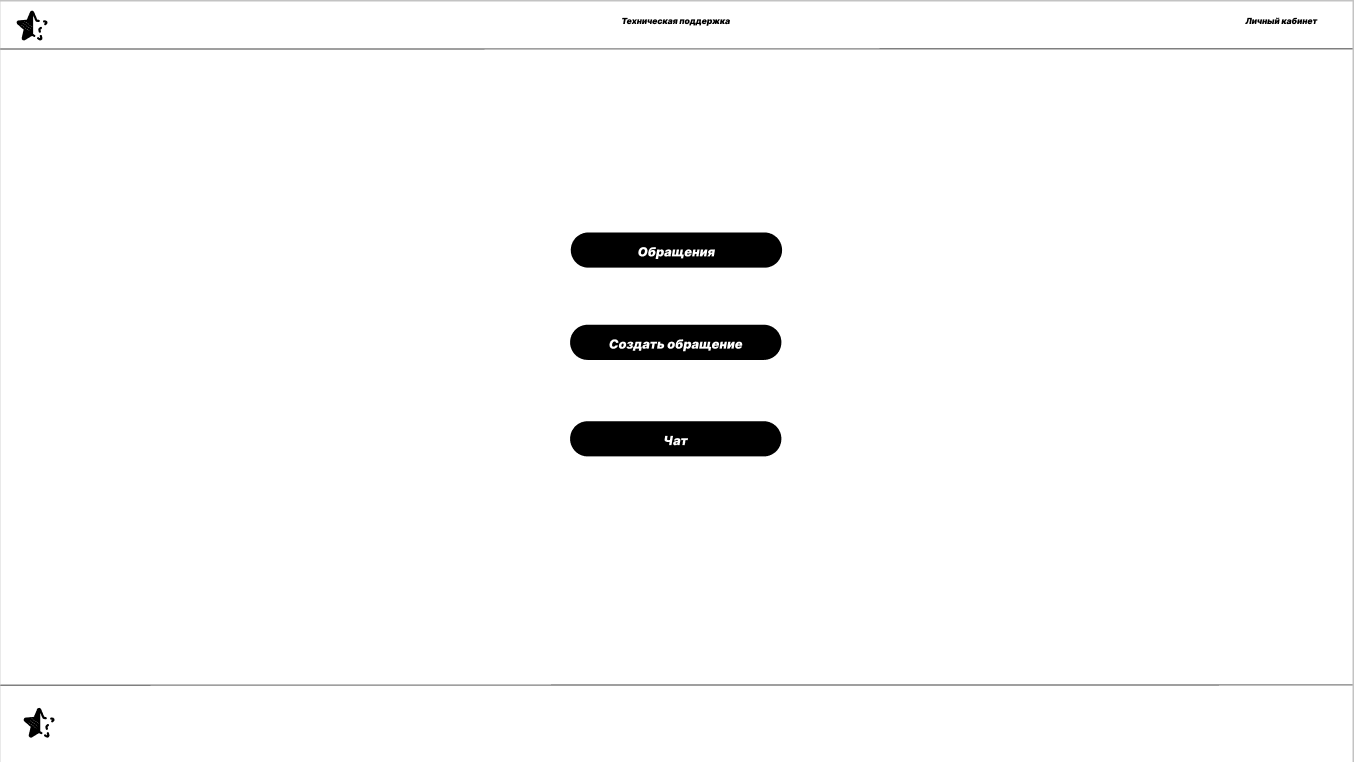


Рисунок 24 – Макет главной страницы

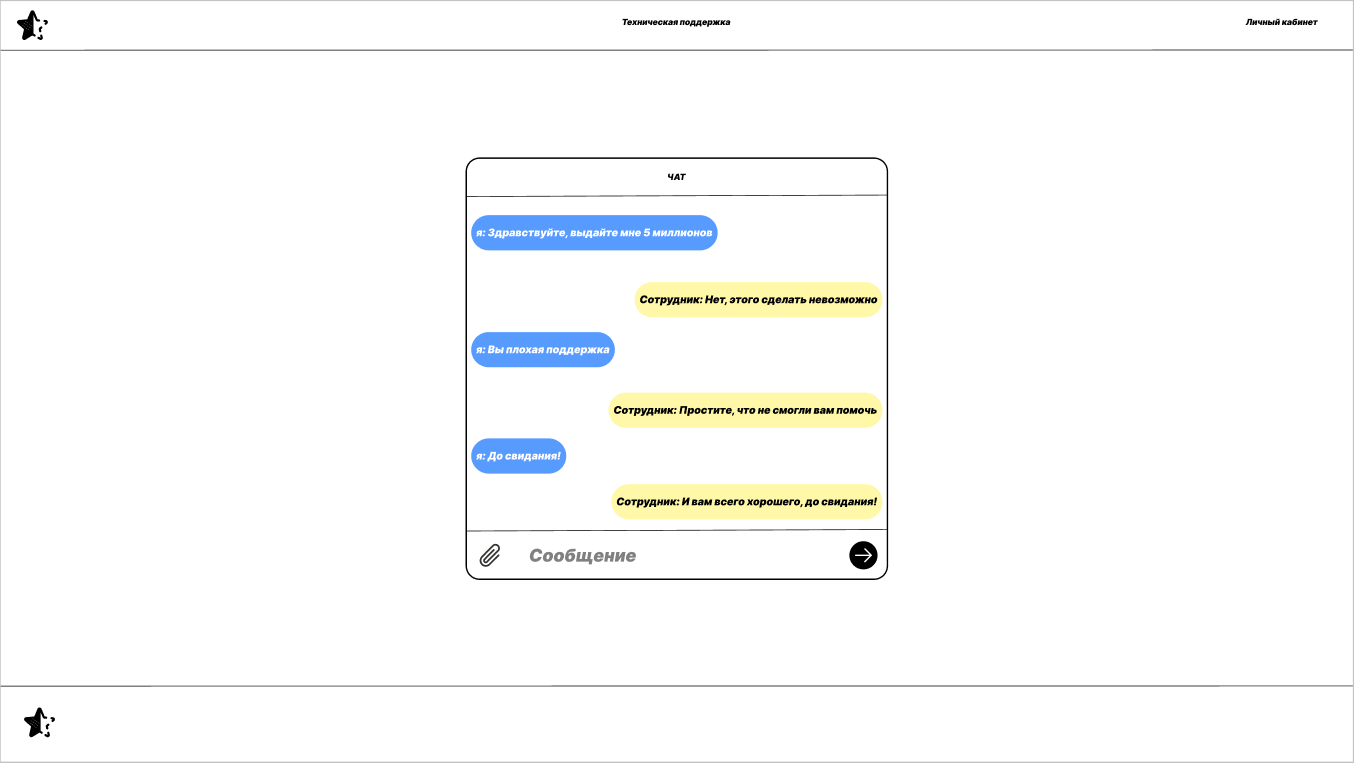


Рисунок 25 – Макет чата

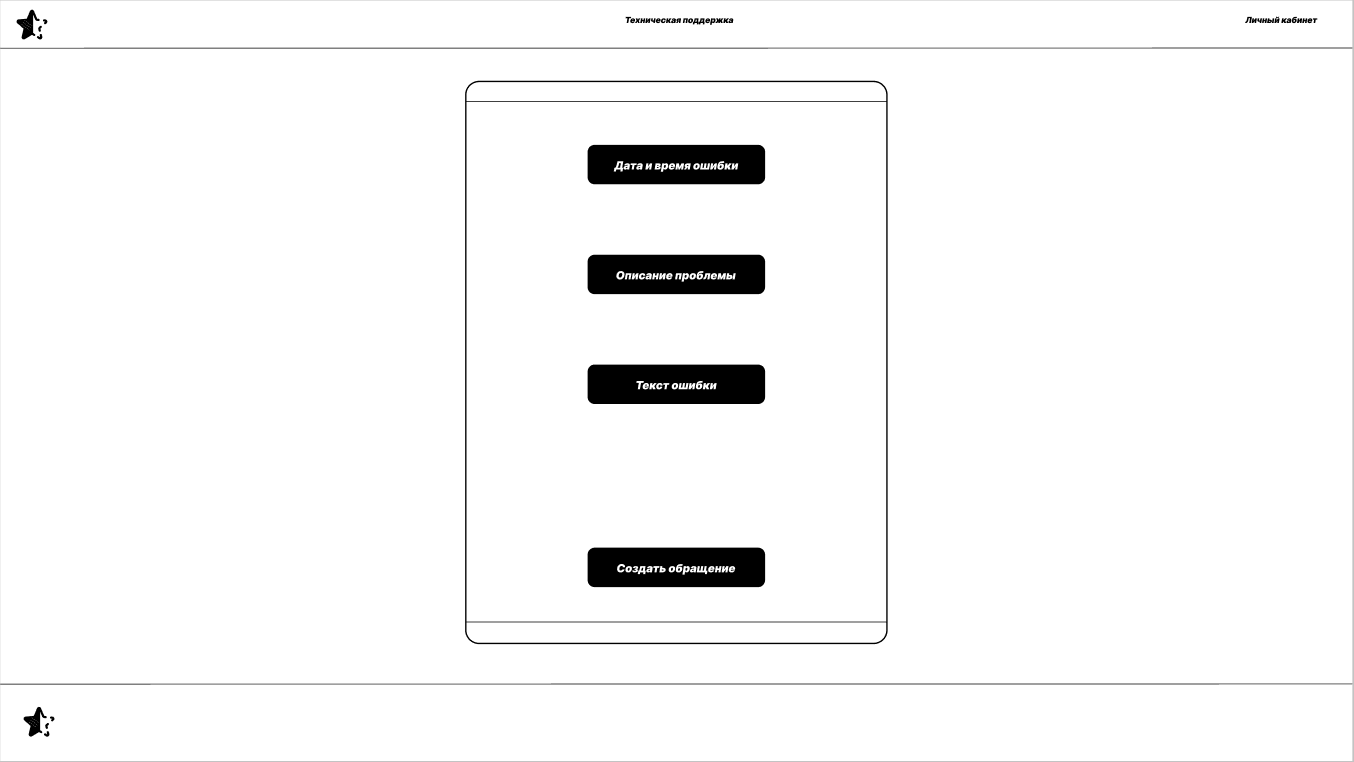


Рисунок 26 – Макет создания обращения

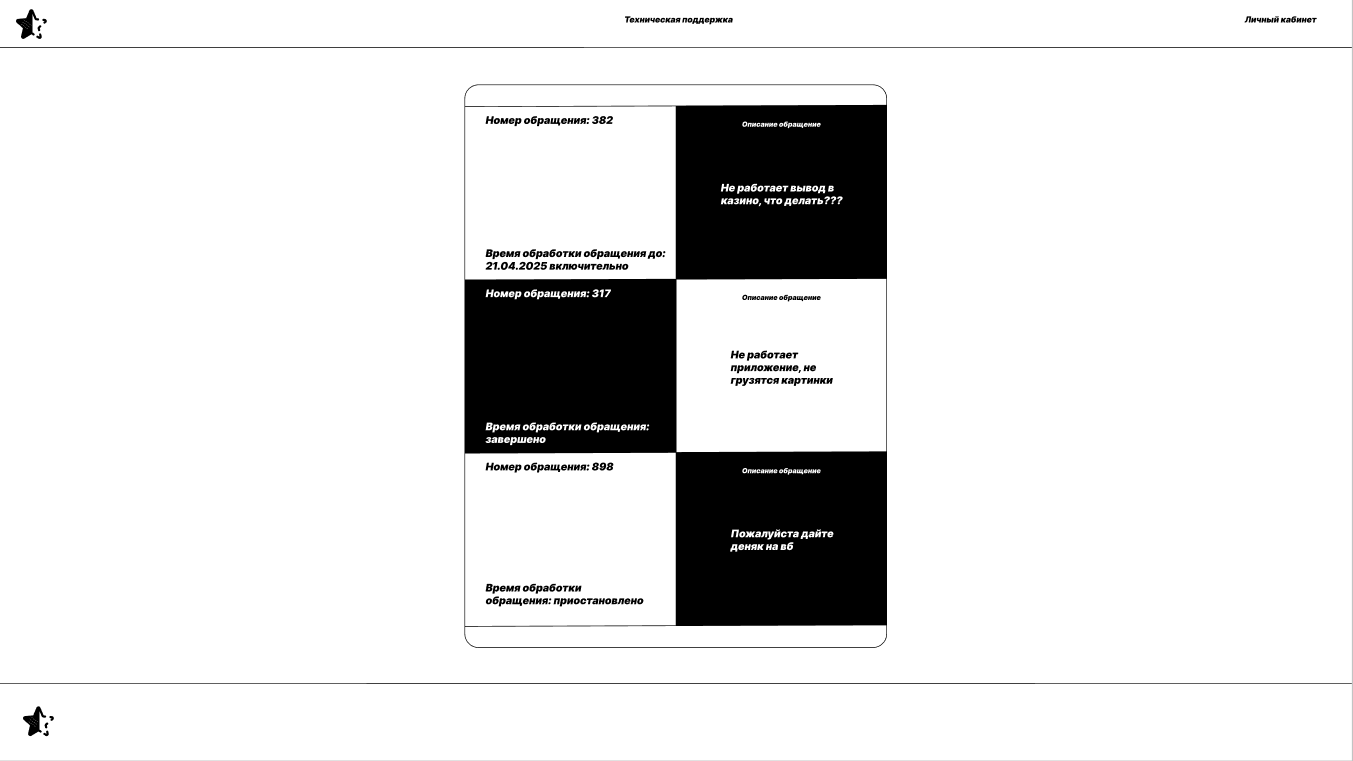


Рисунок 27 – Макет истории обращений

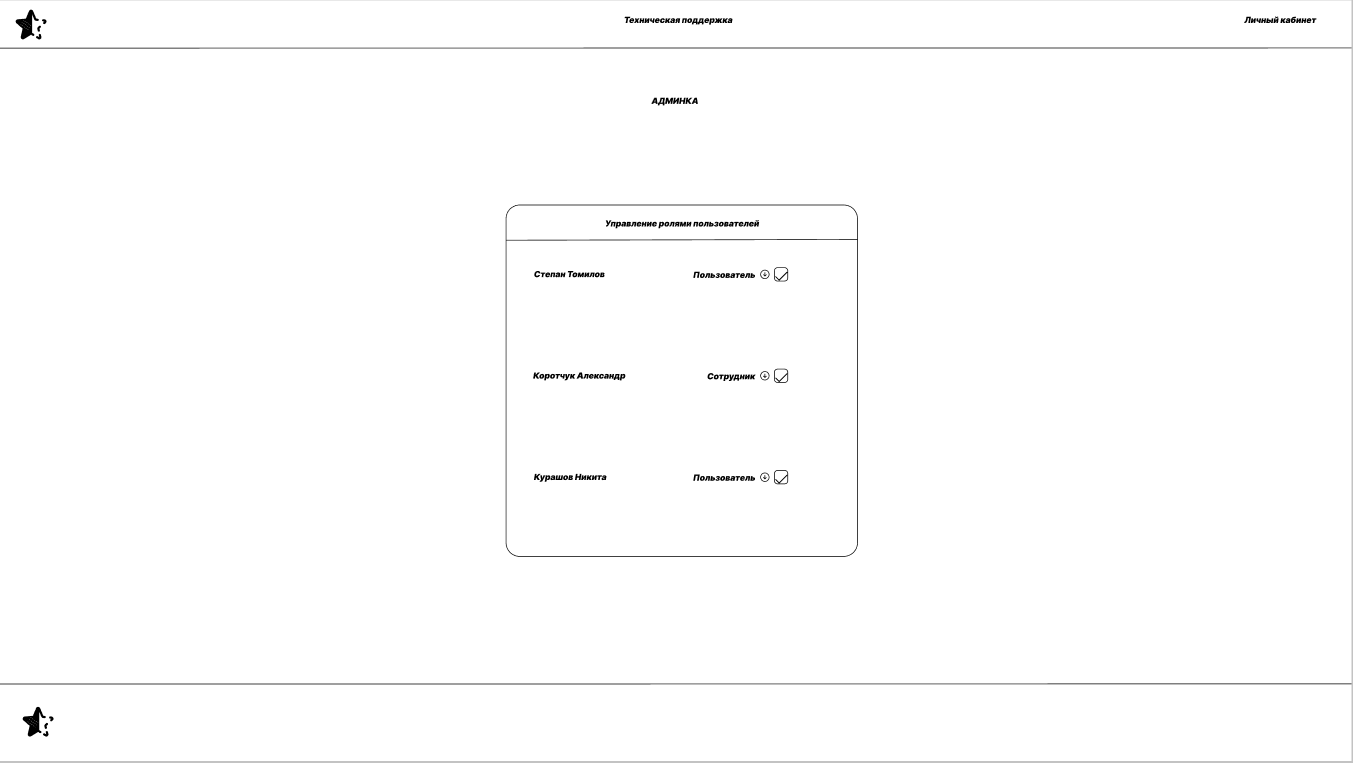


Рисунок 28 – Макет админка

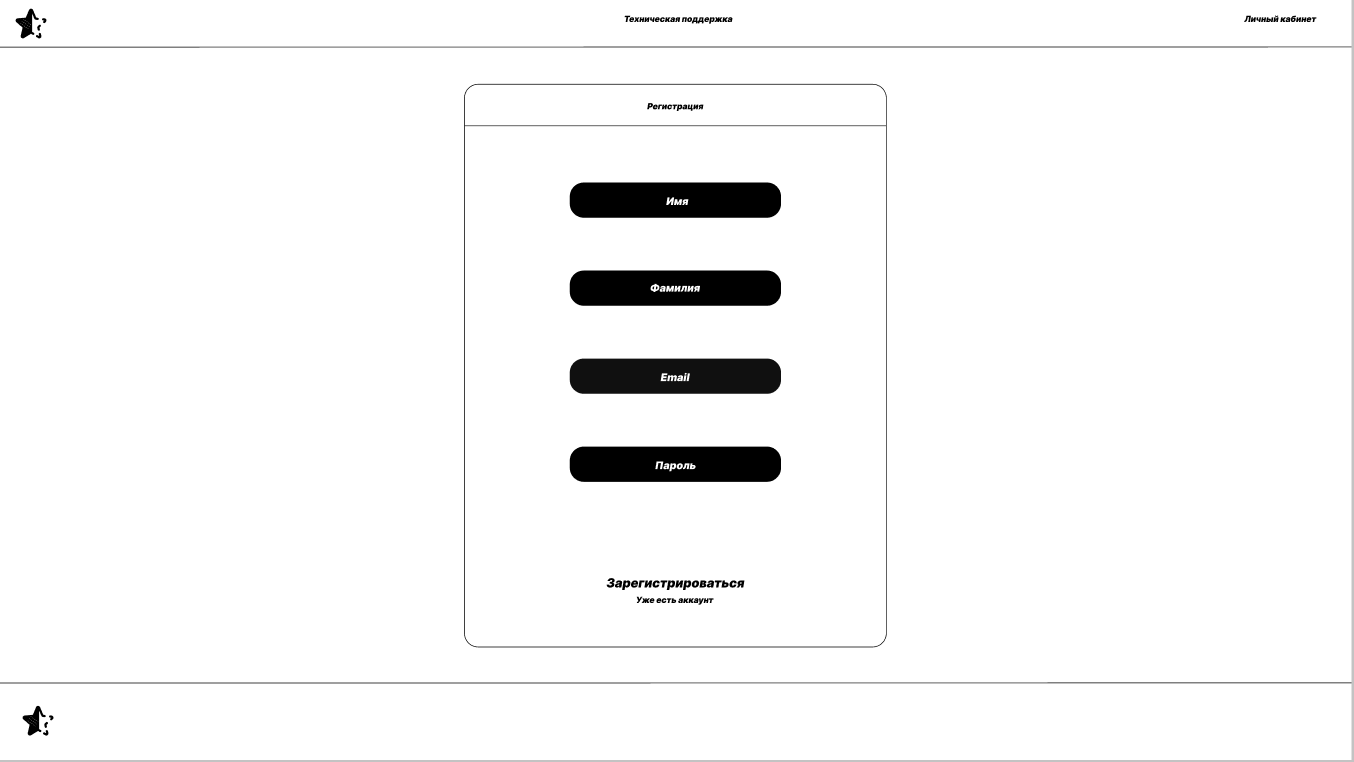


Рисунок 29 – Макет регистрация

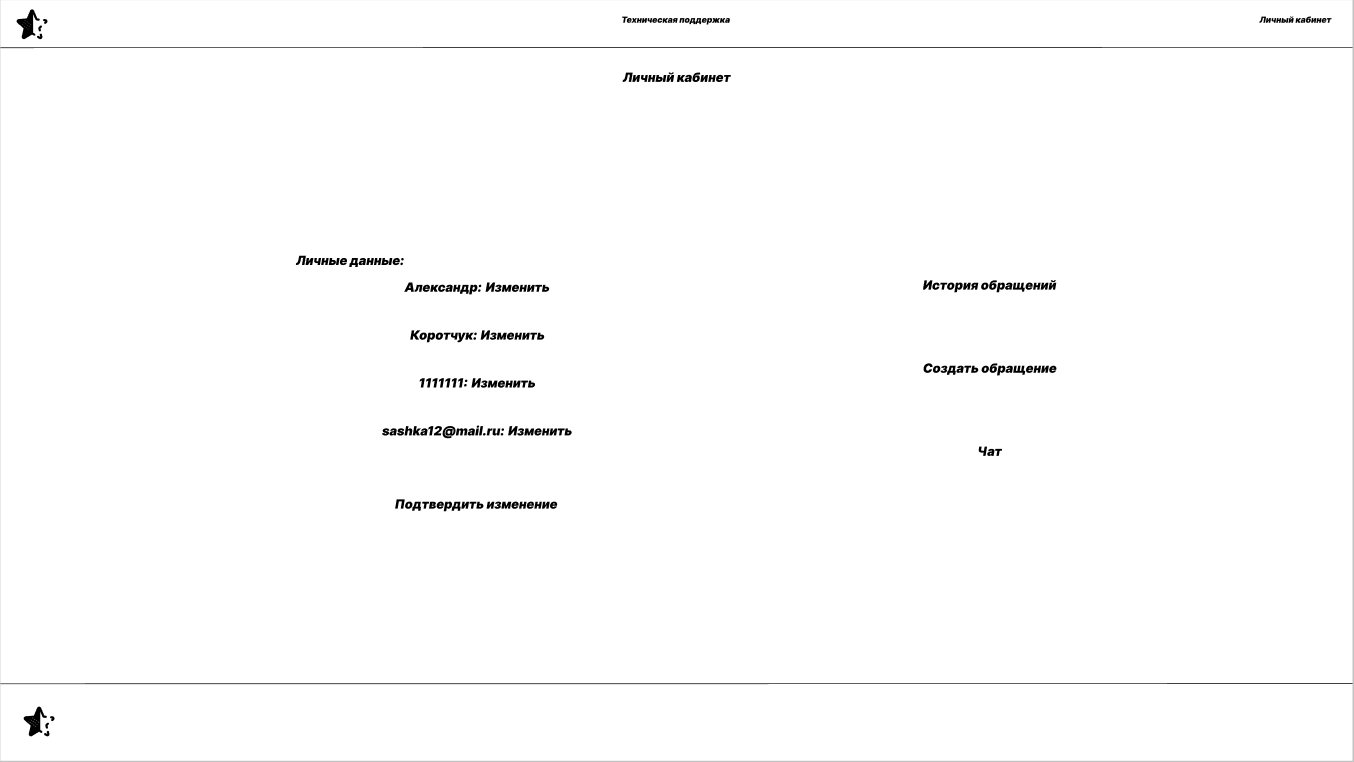


Рисунок 30 – Макет личный кабинет

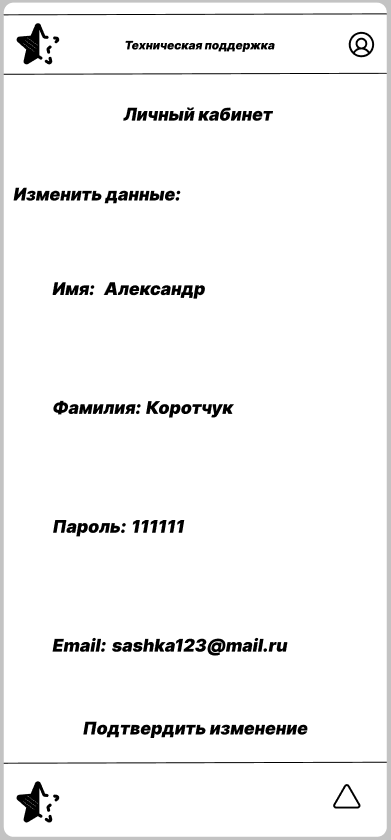


Рисунок 31 – Макет личный кабинет (мобильная версия)

# 4 Разработка веб-приложения

# 4.1 Разработка интерфейса веб-приложения

Разработка пользовательского интерфейса веб-приложения была осуществлена при помощи PHP-фреймворка Laravel и его встроенного шаблона Blade. Он позволяет удобно соединить представления данных и логики приложения. Для создания адаптивного дизайна были применены HTML5, CSS3 и JavaScript.

На рисунках 32/33 представлены две части кода, которые используются в проекте на всех страницах – header и footer. В Header присутствует название системы, логотип и кнопка перехода в личный кабинет. В footer присутствует логотип, а также в адаптивной вёрстке есть кнопка для перехода на главную страницу.



Рисунок 32 – header веб-приложения

****

Рисунок 33 – footer веб приложения

На следующих рисунка 34/35/36 представлены страницы, с которыми чаще всего будут взаимодействовать пользователи системы в зависимости от их ролей.

На рисунке 34 указывается код главной странице, на которой изображены кнопки для перехода на следующие страницы, такие как чат, создать обращение, история обращений и тд. Кнопки могут менять в зависимости от роли пользователя системы.



Рисунок 34 – home.blade

На рисунке 35 предоставлен код, где указана страница создания обращений. На данной странице указаны текстовые поля, где можно подробно всё описать по своей ошибки, добавить текст ошибки, а также зафиксировать возникновение ошибки по времени.



Рисунок 35 – tickets/create.blade

На рисунке 36 предоставлен чат, в котором могут общаться сотрудники поддержки с обычными пользователями для того, чтобы получать быструю обратную связь по ситуации. На странице предоставлен чат с навигацией между диалогами сообщений для сотрудника, при этом блок чата выводит список сообщений, разделяя их на сообщения от пользователя и от поддержки.



Рисунок 36 – chat/index.blade

# 4.2 Разработка базы данных веб-приложения

Второй этап разработки включает в себя создания базы данных. База данных создана с помощью миграций. Каждая миграция создает отдельные таблицы, в которые указываются все необходимые поля, их типы данных, ограничения и связи.

На рисунке 37 представлена комплексная миграция для настройки системы аутентификации. В методе up() создаются три таблицы: основная таблица users содержит базовые поля (id, имя, фамилия, уникальный email, пароль), связана с таблицей ролей через внешний ключ role\_id, включает стандартные поля remember\_token и временные метки; таблица password\_reset\_tokens хранит токены для сброса пароля с привязкой к email и временем создания; таблица sessions фиксирует активные сеансы пользователей с информацией об IP, user-agent и активностью. Метод down() обеспечивает полный откат миграции, удаляя все три таблицы в правильном порядке. Данная миграция формирует полную инфраструктуру для работы системы аутентификации, управления ролями, восстановления паролей и отслеживания сессий.

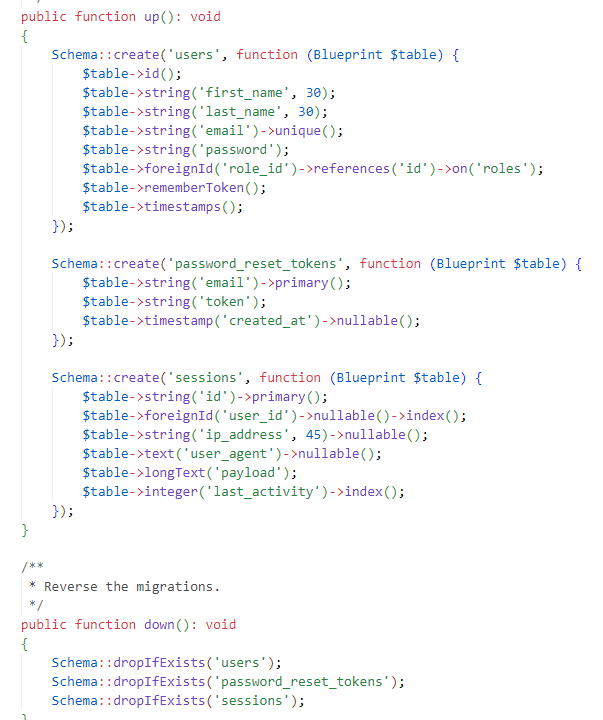


Рисунок 37 – create\_users\_table

На рисунке 38 представлена миграция для создания таблицы tickets в системе. В методе up() определяется структура таблицы обращений: уникальный идентификатор (id), уникальный номер обращения (ticket\_number), связь с пользователем через внешний ключ user\_id, поля description и error\_text для описания проблемы, error\_datetime и processing\_deadline для временных меток ошибки и крайнего срока обработки, связь со статусами через status\_id, а также стандартные поля created\_at и updated\_at. Метод down() обеспечивает удаление таблицы при откате миграции. Эта миграция создает основу для системы обработки обращений, где каждое обращение имеет уникальный номер, привязку к пользователю, подробное описание проблемы, временные параметры и статус обработки, что позволяет организовать полный цикл работы с обращениями пользователей.

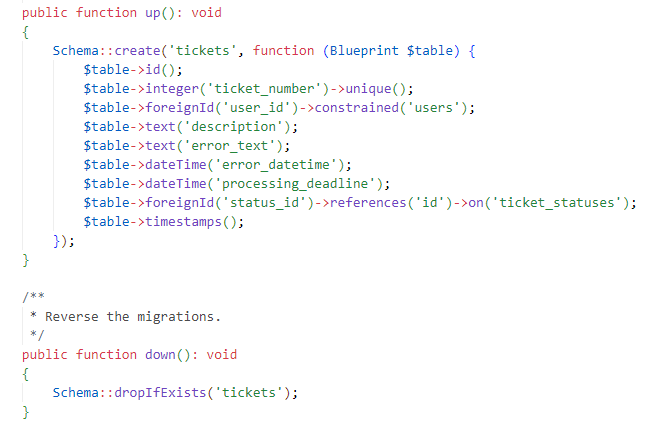


Рисунок 38 – tickets\_table

На рисунке 39 представлена миграция для создания таблицы messages в базе данных. В методе up() определяется структура таблицы: автоматический идентификатор (id), внешние ключи user\_id и support\_id, связывающие сообщения с пользователями (оба ссылаются на таблицу users), поле content для текста сообщения, необязательное поле attachment для вложений, флаг is\_from\_user (по умолчанию true, указывает отправителя), а также стандартные временные метки created\_at и updated\_at. Метод down() обеспечивает откат миграции путем удаления таблицы. Данная миграция создает структуру для чата между пользователями и поддержкой, где каждое сообщение может иметь вложение и четко идентифицировать отправителя.

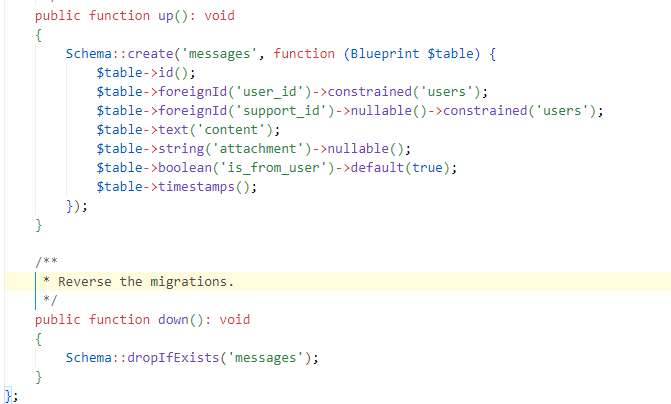


Рисунок 39 – messages\_table

На рисунке 40 представлен UserSeeder, в котором есть информация для заполнения базы данных изначальными пользователями. В данном случае были созданы администратор, сотрудник поддержки и два обычных пользователя. Для каждого пользователя указываются имя, фамилия, email, хешированный пароль и соответствующая роль. Этот сидер обеспечивает первоначальное заполнение системы тестовыми пользователями разных ролей, что полезно, так как не нужно тратить время для регистрации при разработке приложения.



Рисунок 40 – UserSeeder

# 4.3 Разработка серверной части веб-приложения

Крайним этапом разработки является создание серверной части, который обеспечивает обработку логики веб-приложения, взаимодействие с базой данных, управление маршрутами, а также фильтрацию и управление доступами.

В рамках серверной части были разработаны модели, контроллеры, middleware и маршруты. Модели определяют структуру данных и взаимосвязи между сущностями. Контроллеры содержат логику для обработки запросов и формирования ответов. Маршруты задают соответствие между URL-адресами и выполняемыми действиями. Middleware выполняет функции фильтрации и контроля доступа к разделам приложения, интегрируясь через систему маршрутов.

На рисунке 41 представлена модель User. В ней определены важные свойства для работы с пользовательскими данными: массив $fillable указывает поля, которые можно массово назначать (first\_name, last\_name, email, password и role\_id), массив $hidden скрывает конфиденциальные данные (password и remember\_token), а массив $casts автоматически преобразует email\_verified\_at в объект datetime и хеширует пароль при его сохранении. Эти настройки обеспечивают безопасное хранение и обработку пользовательской информации.

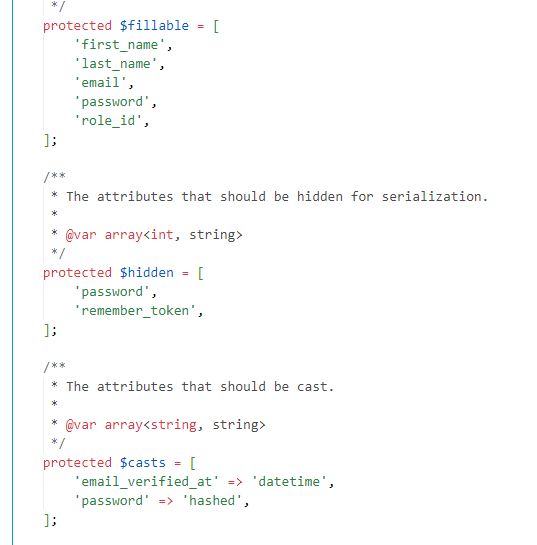
****

Рисунок 41 – модель User

На рисунке 42 представлена Модель Ticket, она представляет собой модель для работы с обращениями в системе. В модели определены заполняемые поля (fillable), включающие номер обращения (ticket\_number), идентификатор пользователя (user\_id), описание проблемы (description), текст ошибки (error\_text), дату возникновения ошибки (error\_datetime), срок обработки (processing\_deadline) и идентификатор статуса (status\_id). Особые поля error\_datetime и processing\_deadline определены в массив $casts с datetime для удобной работы с датами. Модель содержит два отношения: belongsTo к модели User через user\_id для связи с пользователем, создавшим обращение, и belongsTo к модели TicketStatus через status\_id для управления статусами обращений. Данная модель обеспечивает всю необходимую функциональность для хранения и управления данными об обращениях в системе, включая валидацию, преобразование типов и связи с другими сущностями.



Рисунок 42 – модель Ticket

На рисунке 43 представлен UserController, который содержит два основных метода для работы с профилем пользователя. Метод profile() отвечает за отображение страницы профиля текущего авторизованного пользователя, передавая данные пользователя в представление. Метод update() обрабатывает обновление информации профиля: выполняет валидацию входящих данных (проверяет имя и фамилию на обязательность заполнения и длину, email на уникальность и формат, пароль на сложность при его наличии), обновляет соответствующие поля в базе данных, включая хеширование пароля при его изменении, и возвращает пользователя обратно с сообщением об успешном обновлении или ошибками валидации. Вся логика работает исключительно с текущим авторизованным пользователем, получаемым через Auth::user(), что обеспечивает безопасность и предотвращает изменение данных других пользователей.



Рисунок 43 – UserController

На рисунке 44 представлен контроллер, который отвечает за то, чтобы отображать обращения правильному выбранному пользователю по роле, с помощью $selectedUser, а также статус обращения. Продолжение кода TicketController в приложение B.

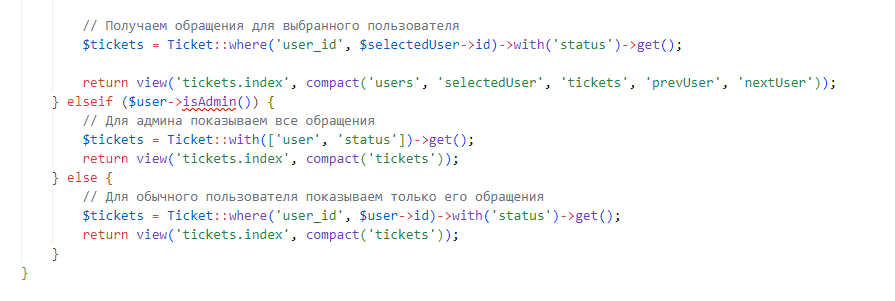


Рисунок 44 – TicketController

На рисунке 45 представлена реализация AdminMiddleware, предназначенного для ограничения доступа к административной панели. В данном промежутке выполняется проверка двух ключевых условий: во-первых, проверяется факт авторизации пользователя в системе, во-вторых, осуществляется проверка соответствия роли пользователя значению "IsAdmin". Только при одновременном выполнении этих условий пользователь получает доступ к админ странице.



Рисунок 45 – AdminMiddleware

Доступ к представлениям реализован через маршруты в файле web.php, где все маршруты сгруппированы по функциональному назначению: аутентификация, защищенные маршруты для авторизованных пользователей, обработка обращений, чат и админ-панель с дополнительной проверкой прав доступа через AdminMiddleware. Приложение использует стандартные HTTP-методы: GET для получения данных и отображения страниц, POST для создания новых записей и PUT для обновления существующих данных. Вся архитектура маршрутизации соответствует схеме на рисунке 46, обеспечивая четкое разделение функционала и контроль доступа к разделам системы.



Рисунок 46 – файл web.php (маршруты)

# 5 Документирования программного продукта

# 5.1 Руководство пользователя веб-приложения для пользователей

Созданный сайт представляет собой веб-приложение технической поддержки. Суть работы веб-приложения заключается в создании обращений обычными пользователями, а решением этих обращений занимаются сотрудники технической поддержки.

Описание установки и запуска проекта:

Для работы на локальном сервере необходимо наличие следующих компонентов, установленных на вашей рабочей системе:

* Composer версии 2.8.3
* PHP версии 8.0 или выше
* MySQL версии 8.0 или выше

Последовательность запуска:

* Открыть терминал в папке проекта.
* Выполнить команду: composer install
* Скопировать файл конфигурации: cp .env.example .env
* Сгенерировать ключ приложения: php artisan key:generate
* Открыть файл .env, указываем настройки для подключения к базе данных: хост, порт, название БД, имя пользователя и пароль.
* Выполнить миграции и заполнить базу данных начальными данными: php artisan migrate:fresh --seed
* Запустить локальный сервер: php artisan serve

После этого веб-приложение будет доступно по адресу:  
http://127.0.0.1:8000

Любой пользователь после перехода по адресной строке будет попадать на страницу входа (рисунок 47), где ему нужно будет ввести его данные для входа, если таковых данных нет, то ему нужно будет нажать на кнопку “создать аккаунт” и перейти на страницу регистрации для регистрации.

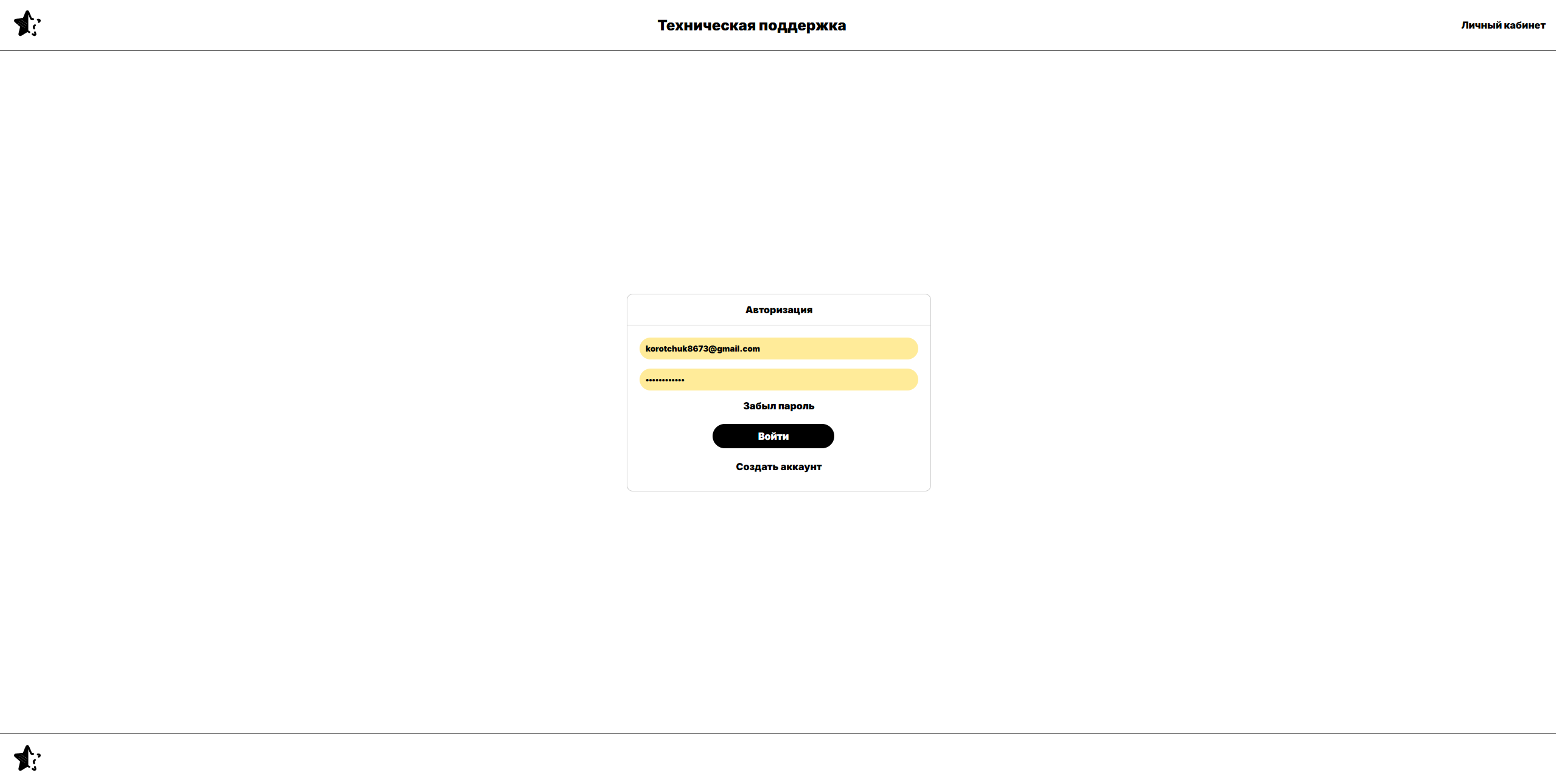


Рисунок 47 – Страница входа

Описание действий пользователя:

* Зарегистрироваться и авторизоваться;
* Заходить чат для того, чтобы написать сотруднику;
* Создавать обращения;
* Просматривать свои обращения;
* Переходить на страницу своего профиля.

Если авторизоваться под сотрудником, то для тебя будут доступны другие действия:

На рисунке 48 указана возможность того, что в чате ты можешь переключаться между пользователями системы, чтобы переходить в разные чаты для ответов, также у тебя есть поиск клиента по его данным.

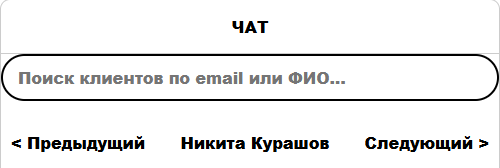


Рисунок 48 – Инструменты поиска для сотрудника

Сотрудник не может создавать обращения, однако может перейти в историю обращений для того, чтобы управлять ими.

На рисунке 49 указано, что сотрудник может обновлять статус обращения (на в обработке, завершено, приостановлено) и его срок его обработки, а также имеет поиск клиента по его данным.

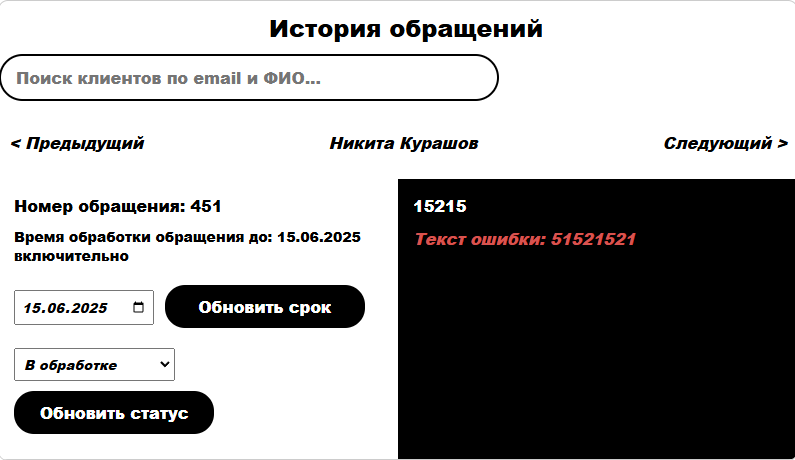


Рисунок 49 – функционал в истории обращений для сотрудника

Также сотрудник имеет возможность зайти в свой личный кабинет для просмотра своих данных.

Если авторизоваться под ролью админа, то для тебя также будет доступен просмотр твоего личного кабинета, однако тебе будет доступна админ-панель. На рисунке 50 изображен вид панели для админа. Админ может искать пользователей системы по их данным, а также менять их роль, например, с сотрудника на пользователя или наоборот, также может выдать роль админа, если это будет нужно.

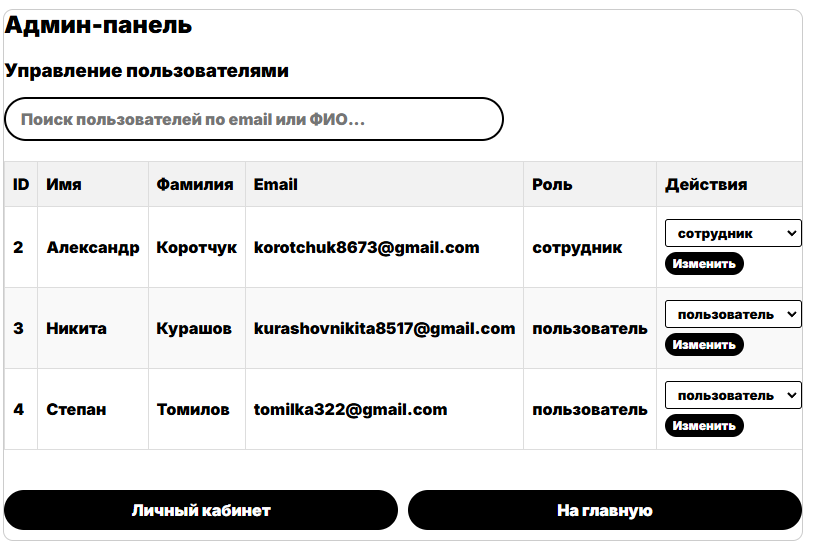


Рисунок 50 – админ-панель

# Заключение

В ходе выполнения курсового проекта было разработано веб-приложение «Техническая поддержка», предназначенное для организации взаимодействия между пользователями и сотрудниками службы поддержки. Приложение обеспечивает удобный интерфейс для создания и обработки обращений, обмена сообщениями в чате, а также управления ролями пользователей (клиент, сотрудник поддержки, администратор). Приложение соответствует поставленным требованиям:

− Обеспечивает удобный и интуитивно понятный интерфейс.

− Поддерживает адаптивный дизайн для мобильных устройств.

− Позволяет эффективно управлять обращениями и коммуникацией между пользователями и поддержкой.

− Включает систему ролей для разграничения прав доступа.

В дальнейшем Разработанное веб-приложение может быть использовано в компаниях для автоматизации работы службы технической поддержки, что повысит качество обслуживания клиентов и оптимизирует внутренние процессы.

# Список используемых источников

1. developer.mozilla.org / Валидация ограничений / URL – <https://developer.mozilla.org/ru/docs/Web/HTML/Guides/Constraint_validation> (Дата обращения 28.03.2025). / – Текст электронный.
2. Figma / Figma помогает командам дизайнеров и разработчиков создавать отличные продукты вместе/ – URL <https://www.figma.com/> (дата обращения: 25.01.2025). – Текст электронный.
3. Github / Intervention Image / URL <https://github.com/Intervention/image> (Дата обращения: 29.03.2025). / – Текст электронный
4. Laravel / Пакеты сообщества / – URL <https://laravel.su/packages?type=file-management> (дата обращения: 16.03.2025). / – Текст электронный.
5. MySQL / Changes in MySQL 8.0.42 (2025-04-15, General Availability) / – URL <https://dev.mysql.com/doc/relnotes/mysql/8.0/en/news-8-0-42.html> (Дата обращения: 27.02.2025). / – Текст электронный.
6. PHP / PHP Manual / URL <https://www.php.net/manual/en/> (Дата обращения 27.02.2025). / – Текст электронный.
7. VS Code / Документация по коду Visual Studio / – URL <https://code.visualstudio.com/docs> (дата обращения: 27.02.2025). – Текст электронный.

# Приложение А

**Министерство образования Иркутской области**

Государственное бюджетное профессиональное

образовательное учреждение Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

**Техническое задание**

**ВЕБ-ПРИЛОЖЕНИЕ «Техническая поддержка»**

|  |  |  |
| --- | --- | --- |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (Н.Р. Карпова) |
|  | (подпись, дата) |  |
| Студент: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | (А.Р. Коротчук) |
|  | (подпись, дата) |  |
|  |  |  |
|  |  |  |
|  |  |  |

Иркутск, 2025

**1. Введение**

**1.1. Общие сведения**

Документ представляет собой техническое задание на создание Технической поддержки. Поддержка будет специализироваться на обработке обращений.

**1.2. Цели и задачи**

Цель создания информационной системы для технической поддержки включает в себя интуитивно понятный интерфейс, удобный процесс заведения обращений, безопасность данных, адаптивность системы, поддержку клиентов. Задачи информационной системы включают:

* Проектирование интерфейса.
* Функциональный чат.
* Форма обращений.
* Личный кабинет.
* Адаптивный дизайн для мобильных устройств.

**2. Основания для разработки**

**2.1. Нормативные документы**

Документ основывается на следующих нормативных документах:

* Приказ №325-у от 24.01.2025 г. Об утверждении тем курсовых проектов.
* Методические указания КП МДК.09.01 2024 ВЕБ.
* ГОСТ 34.602-2020 "Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы".
* ГОСТ Р 56477-2015 "Проектирование и внедрение информационных систем. Общие требования".

**2.2. Проектные документы**

Проектные документы включают:

* Пояснительную записку.
* Руководство пользователя.

**3. Назначение сайта**

**3.1. Общее описание**

Веб-приложение предназначено для организации взаимодействия между пользователями и службой технической поддержки компании. Оно позволяет пользователям отправлять обращения о проблемах, связанных с продуктами или услугами компании, а сотрудники службы поддержки получают возможность оперативно обрабатывать эти запросы.

**3.2. Преимущества и новизна**

Поддержка будет предоставлять:

* Удобный интерфейс.
* Высокую скорость обработки обращений.
* Адаптивность для мобильных устройств.

**4. Требования к Веб-приложению**

**4.1. Функциональные требования**

* Регистрация и авторизация пользователей:
  + Возможность регистраций новых пользователей
  + Вход в систему
  + Восстановление паролей
* Форма со всеми обращениями
  + Добавление обращений
  + Ввод данных для обращений
* Панель для разработчика/администратора:
  + Управление пользователями
  + Управление обращениями
* Личный кабинет
  + Персональные данные
  + История обращений
* Адаптивный дизайн

**4.2. Технические требования**

* Производительность:
  + Время отклика системы не более 0,5 секунд.
  + Обработка до 800 задач одновременно.
* Надежность:
  + Доступность системы не менее 99,9% в год.

**4.3. Эксплуатационные требования**

* Удобство использования:
  + Удобный пользовательский интерфейс.
  + Адаптивный дизайн.

**5. Требования к техническому обеспечению**

**5.1. Оборудование**

* Клиентские рабочие станции: ПК с операционными системами Windows (7, 10, 11), с 4+ ОЗУ.
* Сеть: Доступ в Интернет со скоростью не менее 80 Мбит/с.
* Сетевые протоколы: Поддержка TCP/IP, HTTPS.

**6. Требования к программному обеспечению**

**6.1. Программные компоненты**

* Базы данных: MySQL.
* Программное обеспечение: - Figma, VS code

**6.2. Интерфейсы**

* Интерфейс пользователя: Веб-интерфейс с поддержкой браузеров Chrome, Firefox, Edge, Opera, Yandex.

**7. Организационно-технические требования**

**7.1. Этапы разработки**

В таблице 1 представлены плановые сроки начала и окончания работы по созданию веб-приложения

Таблица 1 – Плановые сроки по созданию веб-приложения

|  |  |
| --- | --- |
| **Наименование этапов курсового проекта** | **Сроки** |
| Предпроектное исследование предметной области (выбор темы, постановка цели, задач, описание области применения, исследование предметной области) | 30.01.25 - 09.02.25 |
| Разработка технического задания (выбор архитектуры программного обеспечения, выбор типа пользовательского интерфейса, выбор языка и среды программирования) | 09.02.25 - 19.02.25 |
| Проектирование программного обеспечения.  (разработка структурной схемы ПО, проектирование базы данных) | 19.02.25 - 04.03.25 |
| Оформление пунктов пояснительной записки (введение, предпроектное исследование, техническое задание и проектирование веб-приложения) | 04.03.25 - 11.03.25 |
| Разработка (программирование) и отладка программного продукта.  Результат- готовый программный продукт. | 11.03.25 - 12.05.25 |
| Составление программной документации (оформление ПЗ, руководство пользователя и презентации) | 13.05.25 - 13.06.25 |

# Приложение B

<?php

namespace App\Http\Controllers;

use App\Models\Role;

use App\Models\Ticket;

use App\Models\TicketStatus;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Validator;

use Illuminate\Support\Facades\DB;

class TicketController extends Controller

{

public function index(Request $request)

{

$user = Auth::user();

if ($user->isSupport()) {

if ($request->has('search')) {

$userRole = Role::where('name', 'пользователь')->first();

$searchTerm = $request->search;

if (empty($searchTerm)) {

$searchResults = User::where('role\_id', $userRole->id)

->orderBy('first\_name')

->limit(10)

->get();

} else {

$searchResults = User::where('role\_id', $userRole->id)

->where(function($query) use ($searchTerm) {

$query->where('email', 'like', '%' . $searchTerm . '%')

->orWhere('first\_name', 'like', '%' . $searchTerm . '%')

->orWhere('last\_name', 'like', '%' . $searchTerm . '%')

->orWhere(DB::raw("CONCAT(first\_name, ' ', last\_name)"), 'like', '%' . $searchTerm . '%');

})

->get();

}

return response()->json($searchResults);

}

$userRole = Role::where('name', 'пользователь')->first();

$users = User::where('role\_id', $userRole->id)->get();

if ($users->isEmpty()) {

return view('tickets.index', ['users' => collect(), 'tickets' => collect()]);

}

$currentUserIndex = 0;

if ($request->has('user\_id')) {

$selectedUser = $users->firstWhere('id', $request->user\_id);

if ($selectedUser) {

$currentUserIndex = $users->search(function($item) use ($selectedUser) {

return $item->id === $selectedUser->id;

});

} else {

$selectedUser = $users->first();

}

} else {

$selectedUser = $users->first();

}

$prevUserIndex = ($currentUserIndex - 1 + $users->count()) % $users->count();

$nextUserIndex = ($currentUserIndex + 1) % $users->count();

$prevUser = $users[$prevUserIndex];

$nextUser = $users[$nextUserIndex];

$tickets = Ticket::where('user\_id', $selectedUser->id)->with('status')->get();

return view('tickets.index', compact('users', 'selectedUser', 'tickets', 'prevUser', 'nextUser'));

} elseif ($user->isAdmin()) {

$tickets = Ticket::with(['user', 'status'])->get();

return view('tickets.index', compact('tickets'));

} else {

$tickets = Ticket::where('user\_id', $user->id)->with('status')->get();

return view('tickets.index', compact('tickets'));

}

}

public function create()

{

if (Auth::user()->isAdmin()) {

return redirect()->route('home')->with('error', 'Администратор не может создавать обращения');

}

return view('tickets.create');

}

public function store(Request $request)

{

if (Auth::user()->isAdmin()) {

return redirect()->route('home')->with('error', 'Администратор не может создавать обращения');

}

$validator = Validator::make($request->all(), [

'error\_datetime' => 'required|date|before\_or\_equal:now',

'description' => 'required|string',

'error\_text' => 'required|string',

], [

'error\_datetime.before\_or\_equal' => 'Дата и время возникновения ошибки не могут быть в будущем',

]);

if ($validator->fails()) {

return redirect()->back()->withErrors($validator)->withInput();

}

$status = TicketStatus::where('name', 'В обработке')->first();

do {

$ticketNumber = rand(1, 999);

} while (Ticket::where('ticket\_number', $ticketNumber)->exists());

Ticket::create([

'ticket\_number' => $ticketNumber,

'user\_id' => Auth::id(),

'description' => $request->description,

'error\_text' => $request->error\_text,

'error\_datetime' => $request->error\_datetime,

'processing\_deadline' => now()->addDays(3),

'status\_id' => $status->id,

]);

return redirect()->route('tickets.index')->with('success', 'Обращение успешно создано');

}

public function updateDeadline(Request $request, Ticket $ticket)

{

if (!Auth::user()->isSupport()) {

return redirect()->back()->with('error', 'У вас нет прав для выполнения этого действия');

}

$validator = Validator::make($request->all(), [

'processing\_deadline' => 'required|date',

]);

if ($validator->fails()) {

return redirect()->back()->withErrors($validator)->withInput();

}

$ticket->processing\_deadline = $request->processing\_deadline;

$ticket->save();

return redirect()->back()->with('success', 'Срок обработки обращения обновлен');

}

public function updateStatus(Request $request, Ticket $ticket)

{

if (!Auth::user()->isSupport()) {

return redirect()->back()->with('error', 'У вас нет прав для выполнения этого действия');

}

$validator = Validator::make($request->all(), [

'status\_id' => 'required|exists:ticket\_statuses,id',

]);

if ($validator->fails()) {

return redirect()->back()->withErrors($validator)->withInput();

}

$ticket->status\_id = $request->status\_id;

$ticket->save();

return redirect()->back()->with('success', 'Статус обращения обновлен');}

}