Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

ПОЛИТЕХНИЧЕСКИЙ КОЛЛЕДЖ

УТВЕРЖДАЮ

Зам. директора по УМ и ВР

Л.Н. Иванова

« » 2025 г.

РАЗРАБОТКА МОДУЛЕЙ ВЗАИМОДЕЙСТВИЯ С ИНФОРМАЦИОННОЙ СИСТЕМОЙ УПРАВЛЕНИЯ НЕЙРОСЕТЯМИ

Пояснительная записка к дипломному проекту по специальности

09.02.07 Информационные системы и программирование

ПТК. ДП 1991 25. 000ПЗ

|  |  |
| --- | --- |
| Согласовано: |  |
| Консультант по спец. части  Л.Н. Цымбалюк  « » 2025 года  Консультант по экон. части  Г.В. Лебедева  « » 2025 года  Нормоконтроль  А.М. Чернега  « » 2025 года  Заместитель директора по УПР  А.М. Чернега  « » 2025 года | Руководитель  Д.А. Смородин  « » 2025 года  Выполнил:  обучающийся группы 1991  Д.С. Шустович  « » 2025 года |

Содержание

[Введение 3](#_Toc200228726)

[1 Общая часть 5](#_Toc200228727)

[1.1 Постановка задачи 5](#_Toc200228728)

[1.2 Обоснование проектных решений 11](#_Toc200228729)

[1.3 Обзор и анализ существующих программных систем 14](#_Toc200228730)

[2 Специальная опытно-экспериментальная часть 18](#_Toc200228731)

[2.1 Анализ задачи 18](#_Toc200228732)

[2.2 Описание логической структуры 24](#_Toc200228733)

[2.3 Описание работы программы 28](#_Toc200228734)

[2.4 Руководство оператора 31](#_Toc200228735)

[3 Экономическая часть 35](#_Toc200228736)

[3.1 Расчёт заработной платы с отчислением на социальное страхование 35](#_Toc200228737)

[3.2 Расчёт стоимости материалов и лицензионного обеспечения 40](#_Toc200228738)

[3.3 Расчёт накладных расходов 42](#_Toc200228739)

[3.4 Составление и расчёт цены реализации программного продукта 43](#_Toc200228740)

[Заключение 45](#_Toc200228741)

[Список литературы 46](#_Toc200228742)

[Приложение А 47](#_Toc200228743)

[Приложение Б 57](#_Toc200228744)

[Приложение В 62](#_Toc200228745)

[Приложение Г 65](#_Toc200228746)

# Введение

В наше время технологии, которые используют нейросети, активно развиваются и находят применения в самых разных областях, например, в медицине, финансах, промышленности и информационных технологиях. Для более эффективного управления ими необходимо разработать решение, которое позволит автоматизировать создание, обучение и управления нейросетями, а также упростит анализ данных и интеграцию в различные сервисы.

Актуальность работы обусловлена ростом объема данных и необходимостью оперативной обработки информации с использованием нейросетевых моделей. Разработка модулей взаимодействия для информационной системы управления нейросетями позволит автоматизировать управление, создание и обучение нейросетей, а также упростит анализ данных и добавит возможность интегрировать нейросети в различные сервисы.

Целевой аудиторией будут являться разработчики и бизнес-аналитики. Для разработчиков система обеспечит удобный интерфейс для взаимодействия с нейросетевыми моделями. Для бизнес-аналитиков система будет полезна возможностью анализа данных, прогнозирования трендов и оптимизации процессов.

Целью дипломной работы является разработка модулей взаимодействия с информационной системой управления нейросетями, обеспечивающих возможность автоматизировать создание, обучение и управление нейросетевыми моделями, а также облегчить анализ данных и добавить возможность интеграции нейросетей в различные сервисы.

Для достижения поставленной цели, были выделены следующие задачи:

* изучить существующие методы и технологии для работы с нейросетями;
* спроектировать базу данных, для хранения информации об аккаунтах, рабочих областях, внутренних пользователей, ролей, прав пользователей, хостингах и интеграций;
* разработать сервер для взаимодействия с базой данных и с информационной системой управления нейросетями;
* разработать дизайн для модулей взаимодействия;
* разработать модули взаимодействия по заготовленному дизайну;
* провести тестирование и отладку модулей взаимодействия и сервера.

Объектом исследования являются процессы взаимодействия модулей с нейросетевыми моделями, включающих обработку данных, создание, обучение и управление моделями, а также их интеграцию в различные сервисы. Предметом исследования будет являться исследование методов и программных средств, обеспечивающих взаимодействие с информационной системой управления нейросетями.

Методами исследования, будут являться:

* проектирование баз данных для хранения и обработки информации;
* разработка программного обеспечения с использованием методологий объектно-ориентированного и функционального программирования;
* тестирование и отладка модулей взаимодействия и сервера.

Практическая значимость дипломного проекта заключается в создании модулей взаимодействия с информационной системой управления нейросетями, которые:

* автоматизируют работу с нейросетевыми моделями;
* гибкая интеграция нейросетей в сервисы;
* повышают качество анализа и обработки больших данных.

Разработанные модули взаимодействия могут быть полезны для автоматизации работы с нейросетями в различных сферах, таких как бизнес-аналитика, медицинская диагностика, обработка изображений и другие.

# 1 Общая часть

# 1.1 Постановка задачи

1.1.1 Обоснование необходимости разработки

В соответствии с заданием на выпускную квалификационную работу, требуется разработать модули взаимодействия с информационной системой управления нейросетями для эффективного и надёжного взаимодействия с нейросетевыми моделями

Требования к производительности, надёжности и масштабируемости современных информационных систем постоянно растут, разработка новых модулей, соответствующих этим требованиям, позволит создать систему, способную эффективно обрабатывать большие объёмы данных и обеспечивать высокую доступность и надёжность работы.

Таким образом, разработка модулей взаимодействия для информационной системы управления нейросетями поможет решить актуальные задачи, что позволит улучшить взаимодействие с нейросетевыми моделями и обеспечить их интеграцию с различными сервисами.

1.1.2 Технико-математическое описание задачи

Разработка модулей взаимодействия будет осуществляться с использованием комплекса современных технологических решений и методических подходов.

В процессе создания системы задействовались прогрессивные инструменты программирования и проверенные архитектурные принципы. Особое внимание уделяется выбору технологий, обеспечивающих стабильную работу и лёгкую масштабируемость проекта.

Используемые технологии и методы:

* HTTP/HTTPS – протоколы передачи данных, которые используются в современном интернет-взаимодействии, данная технология обеспечивает надёжный обмен информацией между клиентскими устройствами и серверами, формируя основу для работы веб-приложений и сайтов;
* SQL – язык структурированных запросов, он предназначен для работы с реляционными базами данных – массивами информации, которые связаны между собой и представлены в виде таблиц;
* postgreSQL – система управления базами данных, которая сочетает в себе принципы реляционной и объектно-ориентированной модели, будучи объектно-реляционной, СУБД поддерживает не только традиционные табличные структуры данных, но и более сложные объектные конструкции, что значительно расширяет её функциональные возможности;
* HTML – язык разметки гипертекста, который лежит в основе создания веб-страниц, этот стандартизированный язык определяет структуру и содержание веб-документов с помощью специальных элементов, и тегов, а также каждый HTML-документ представляет собой иерархическую систему элементов, которые браузер интерпретирует и отображает в виде визуальной веб-страницы;
* CSS – язык стилей, который используется для оформления внешнего вида веб-страниц, с помощью данного инструмента можно задать цвета, шрифты, расположение элементов и другие визуальные характеристики интерфейса;
* javascript – основной язык для разработки интерактивных веб-приложений, этот высокоуровневый язык программирования позволят добавлять динамическое поведение на веб-страницы, превращая статичный контент в полноценные интерактивные приложения, также благодаря своей интеграции со всеми современными браузерами JavaScript стал фундаментальной технологией, позволяя обрабатывать действия пользователя в реальном времени без перезагрузки страницы;
* typescript – представляет собой мощное расширение для JavasScript, добавляющее строгую статическую типизацию, основное предназначение этой надстройки, это повышение надёжности и удобства поддержки кода за счёт введения системы типов, которая помогает выявлять потенциальные ошибки ещё на этапе разработки, до выполнения кода в браузере или Node.js.
* vue.js – современный и прогрессивный JavaScript-фреймворк разработанный для эффективного создания сложных пользовательских интерфейсов, этот инструмент позволяет постепенно внедрять свои возможности в проект, начиная от простого подключения на одной странице, до построения полноценный одностраничных приложение, в дипломном проекте, фреймворк используется как основной инструмент для построения одностраничного веб-приложения;
* vue-router – библиотека маршрутизации для фреймворка Vue.js, которая предоставляет мощный инструмент для организации навигации в одностраничных приложениях, эта библиотека интегрируется с экосистемой Vue, позволяя создавать сложные маршрутизируемые интерфейсы с сохранением всех преимуществ реактивности и компонентного подхода, основной принцип работы заключается в динамическом отображении различных компонентов Vue в зависимости от URL-адреса в браузере, при этом страница не перезагружается полностью, что создаёт пользовательский опыт, характерный для современных веб-приложений;
* pinia – представляет собой современную библиотеку для управления состоянием приложений, разработанная для фреймворка Vue.js, этот инструмент предлагает простой и интуитивно понятный способ централизованного хранения и управления данными, которые должны быть доступны в различных компонентах приложения, в отличии от традиционных подходов, Pinia обеспечивает реактивность состояния, автоматически отслеживая изменения данных и синхронизируя их между всеми частями приложения, что исключает необходимость ручного обновления интерфейса при изменении данных;
* nest.js – современный фреймворк для разработки эффективных и масштабируемых серверных приложений на Node.js, этот инструмент сочетает в себе элементы объектно-ориентированного, функционального и реактивного программирования, предлагая структурированный подход для создания масштабируемых серверных приложений в отличие от многих других Node.js-фреймворков, Nest.js предоставляет готовую архитектуру приложения, что позволяет сразу начинать разработку с продуманной структурой;
* node.js – кроссплатформенная среда выполнения JavaScript, которая позволяет использовать его для серверной разработки, в основе данной среды выполнения лежит высокопроизводительный движок V8, который компилирует JavaScript в машинный код, обеспечивая исключительную скорость выполнения;
* REST API - представляет собой широко распространённый подход к организации взаимодействия между клиентским приложениями и серверами, основанный на принципах архитектурного стиля REST, в основе этого подхода лежит использование стандартных HTTP-методов (GET, POST, PUT, DELETE и других) для выполнения операций с ресурсами, которые идентифицируются через URI;
* инструменты для тестирования – в качестве основного решения для модульного и интеграционного тестирования как клиентской, так и серверной частей приложения выбран фреймворк Jest, а для комплексного end-to-end тестирования пользовательского интерфейса будет использован инструмент Cypress, позволяющий моделировать реальные пользовательские сценарии и проверять приложение в условиях, максимально приближенным к рабочему сценарию;
* git – представляет собой распределённую систему управления версиями, которая будет использоваться в дипломном проекте для эффективного контроля изменений исходного кода.

1.1.3 Характеристика бизнес-процессов

Модули взаимодействия разработаны для обеспечения стабильной и производительной интеграции с информационной системой управления нейросетями. В архитектуре модулей предусмотрены механизмы оперативного обновления информации в реальном времени, включая текущее состояние моделей, аналитику их работы и показатели интеграции с внешними сервисами. Это позволяет поддерживать актуальность данных и оперативно реагировать на изменения в информационной системе управления нейросетями.

Для структурированного представления и глубокого анализа бизнес-процессов проекта, применяется методология IDEF0, дающая возможность наглядно отображать функциональные взаимосвязи через иерархические модели с различными уровнями детализации. В рамках этой методологии выделены ключевые бизнес-процессы, охватывающие весь жизненный цикл взаимодействия с информационной системой управления нейросетями – от создания хостинга и его конфигурации до мониторинга работы нейросетей и обработки запросов от сторонних сервисов, у которых есть интеграция с модулями взаимодействия. Такой подход обеспечивает комплексное понимание всех аспектов работы системы и позволят оптимизировать процессы на каждом уровне взаимодействия.

Основные бизнес-процессы:

* управление хостингами;
* получение статистики работы нейросетей;
* создание интеграций для сторонних сервисов;
* получение статистики работы интеграции;
* управление пользователями;
* управление ролями;
* управление правами пользователей и ролей.

На рисунке А.1 представлена IDEF0 модели настройки прав пользователя. На рисунке А.2 продемонстрирована декомпозиция IFED0 модели настройки прав пользователя, которая включает шаги: выбор пользователя, изменение прав пользователя, сохранение изменённых прав, вывод сообщения об успешном сохранении прав пользователя. Также была создана диаграмма прецедентов, продемонстрированная на рисунке А.3.

Пользователи системы и их функционал:

* администраторы:
  1. авторизация и регистрация;
  2. просмотр, создание, удаление и управление пользователями и модераторами;
  3. управление правами пользователей и модераторов;
  4. управление правами ролей;
  5. просмотр, создание, удаление и управление хостингами;
  6. просмотр, создание, удаление и управление интеграциями;
  7. управление и возможность удалить рабочее пространство.
  8. управление аккаунтом.
* модераторы:
  1. авторизация и регистрация;
  2. создание и управление хостингами;
  3. создание и управление интеграциями;
  4. просмотр, создание, удаление и управление пользователями;
  5. управление правами пользователей;
  6. управление аккаунтом.
* пользователи:
  1. авторизация и регистрация;
  2. просмотр хостингов;
  3. просмотр сервисов;
  4. управление аккаунтом.

Внедрение данной системы направлено на оптимизацию процессов взаимодействия с нейросетевыми моделями и повышения эффективности их управления. Такой подход обеспечит существенное улучшение качества собираемых данных за счёт автоматизации ключевых операций и снижения потенциальных ошибок, связанных с человеческим фактором.

1.1.4 Требования к программе

Для корректной работы системы сервер должен соответствовать следующим минимальным техническим требованиям:

* операционная система – Ubuntu 20 и выше;
* процессов Intel i3 или выше;
* оперативная память – от 4 Гб;
* свободное место на диске – от 15 Гб для разворачивания программы и последующего хранения поступающих данных.

## 1.2 Обоснование проектных решений

1.2.1 Обоснование выбора языков программирования

Для реализации модулей взаимодействия был выбран JavaScript как основной язык программирования. Данный язык обеспечивает кроссплатформенную совместимость, и позволяет разрабатывать универсальные решения, работающие как на стороне клиента, так и на сервере через Node.js, а также обладает рядом ключевых преимуществ:

* typescript: Добавляет в разработку статическую типизацию, что существенно уменьшает вероятность возникновения ошибок во время выполнения программы и способствует созданию более надежного кода, также благодаря строгой проверке типов на этапе компиляции, разработчики могут выявлять и исправлять потенциальные проблемы ещё до запуска приложения;
* универсальность: JavaScript позволяет разрабатывать как внешние интерфейсы, так и серверные приложения, в рамках одной экосистемы, что упрощает поддержку кода;
* поддержка асинхронного программирования: JavaScript позволяет выполнять большие и сложные операции, без блокирования основного потока, что критически важно при большой обработке данных;
* экосистема и сообщество: JavaScript имеет очень большое сообщество разработчиков, что даёт возможность быстро решать возникающие ошибки или находить материал по нужной тематике.

Таким образом, выбор JavaScript для разработки модулей взаимодействия определена его возможностями для создания масштабируемых и кроссплатформенных веб-приложений с возможностью иметь одну кодовую базу, как и на внешнем интерфейсе приложения, так и на его серверной части.

1.2.2 Инструментальные средства

В рамках разработки модулей взаимодействия будет задействован следующие комплекс современных инструментальных средств:

* figma – современный графический редактор для создания пользовательских интерфейсов, интерактивных прототипов и любых дизайн-проектов, этот мощный инструмент сочетает в себе широкие возможности векторного редактирования с удобными функциями для проектирования UI/UX, позволяя разрабатывать сложные интерфейсные решения с пиксельной точностью;
* postgreSQL – объектно-реляционная система управления базами данных, выбрана за её возможность легко масштабировать проекты и стабильность работы, используется для хранения информации об аккаунтах, рабочих областях, внутренних пользователей, ролей, прав пользователей, хостингах и интеграций с сервисами.

Библиотеки:

* day.js – представляет собой компактную и производительную JavaScript-библиотеку, предназначенную для обработки, форматирования манипуляции с датами и временем, эта легковесная библиотека схожа с Moment.js, но существенно меньше по размерам пакета, что делает её идеальным выбором для современных веб-приложений, где важна оптимизация производительности, также данная библиотека поддерживает все основные операции с датами: парсинг из различных форматов, валидация, вычисления между датами, локализацию и сложные манипуляции с временными промежутками;
* underscore.js – предоставляет набор утилитарных функций для эффективной работы с различными структурами данных, она содержит много полезных методов, упрощающих манипуляции с массивами, коллекциями, объектами и функциями, что делает код более читаемым и выразительным;
* jest – фреймворк для написания модульных тестов на JavaScript, позволяющий корректировать работы отдельных модулей программы и минимизировать ошибки;
* cypress – современный инструмент для сквозного end-to-end тестирования веб-приложений.

Выбор данных инструментальных средств, основан на их популярности, сообществом разработчиком. Они обеспечивают высокую производительность, безопасность и удобство разработки модулей взаимодействия с информационной системой управления нейросетями.

1.2.3 Обоснование выбора среды программирования

В качестве среды разработки выбрана Visual Studio Code, т.к. она обладает рядом факторов, обеспечивающих эффективную разработке:

* Популярность: Visual Studio Code, является одним из самых популярных и функциональных инструментов для разработки на языке JavaScript, а благодаря большому количеству разработчиков, использующих её, можно очень просто найти нужный плагин или легко настроить редактор под свой проект;
* Поддержка системы контроля версий: Visual Studio Code может поддерживать Git с помощью плагинов, а также имеет возможность отслеживать внесённые изменения в кодовую базу проекта;
* Отладка проекта: Встроенные средства помогают настроить отладку проекта, что в разы ускоряет устранение возникающих ошибок и ускоряет разработку.

1.2.4 Информационное обеспечение

Для реализации модулей взаимодействия с информационной системой управления нейросетями выбраны следующие программные средства и инструменты:

* postgreSQL – представляет собой объектно-реляционную систему управления базами данных, которая была выбрана для дипломного проекта, благодаря своему уникальному сочетанию надежности, функциональности и гибкости, в отличие от многих других СУБД, PostgreSQL предлагает не только классические реляционные возможности, но и поддержку объектно-ориентированных подходов, что делает её очень подходящим решением для разработки модулей взаимодействия;
* git – распределённая система контроля версий, которая была выбрана для проекта благодаря своей надежности и гибкости;
* diagrams.net – кроссплатформенное приложение для построения диаграмм и схем, которое было выбрано для проектирования архитектуры приложения благодаря своей универсальности и богатому функционалу.

# 1.3 Обзор и анализ существующих программных систем

В рамках анализа существующих программных систем были рассмотрены системы, позволяющие управление и интеграцию нейросетей. Особое внимание уделялось возможностям системы.

Сервис Yandex Cloud характеризуется полной поддержкой TenserFlow, PyTorch, CatBoost, а также API для NLP и компьютерного зрения. Пользователю доступна функция бесплатного периода и кредиты для старта, также интерфейс очень похож на AWS/GCP, что существенно облегчает переход разработчиков над данный сервис. Также у сервиса можно развернуть GPU для обучения моделей. Однако, данный сервис является дороже своих аналогов, также имеет маленькую интеграцию с корпоративными системами и имеет очень высокую зависимость от экосистемы Яндекса.

Рисунок 1 – Главная страница Yandex Cloud

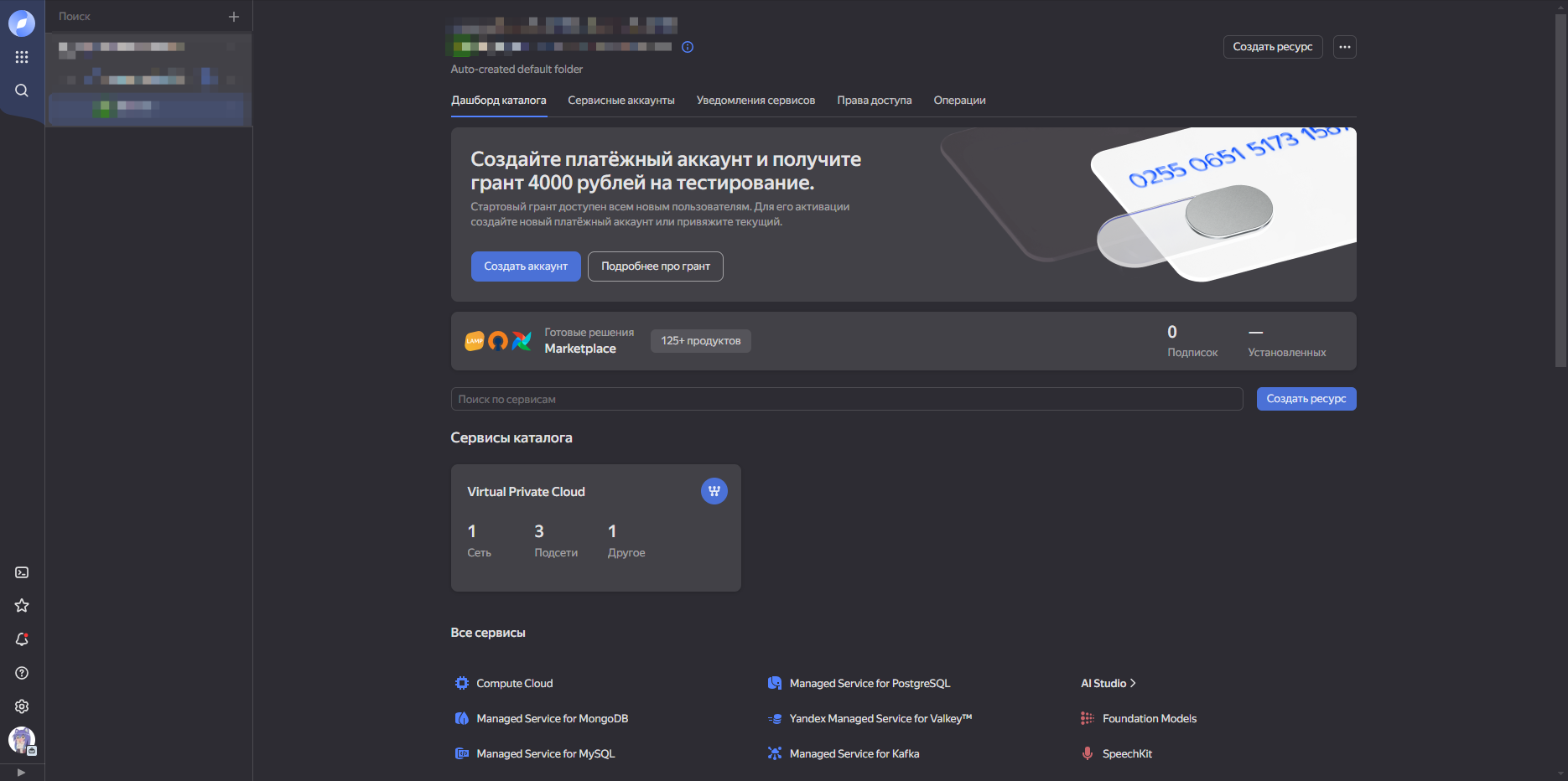
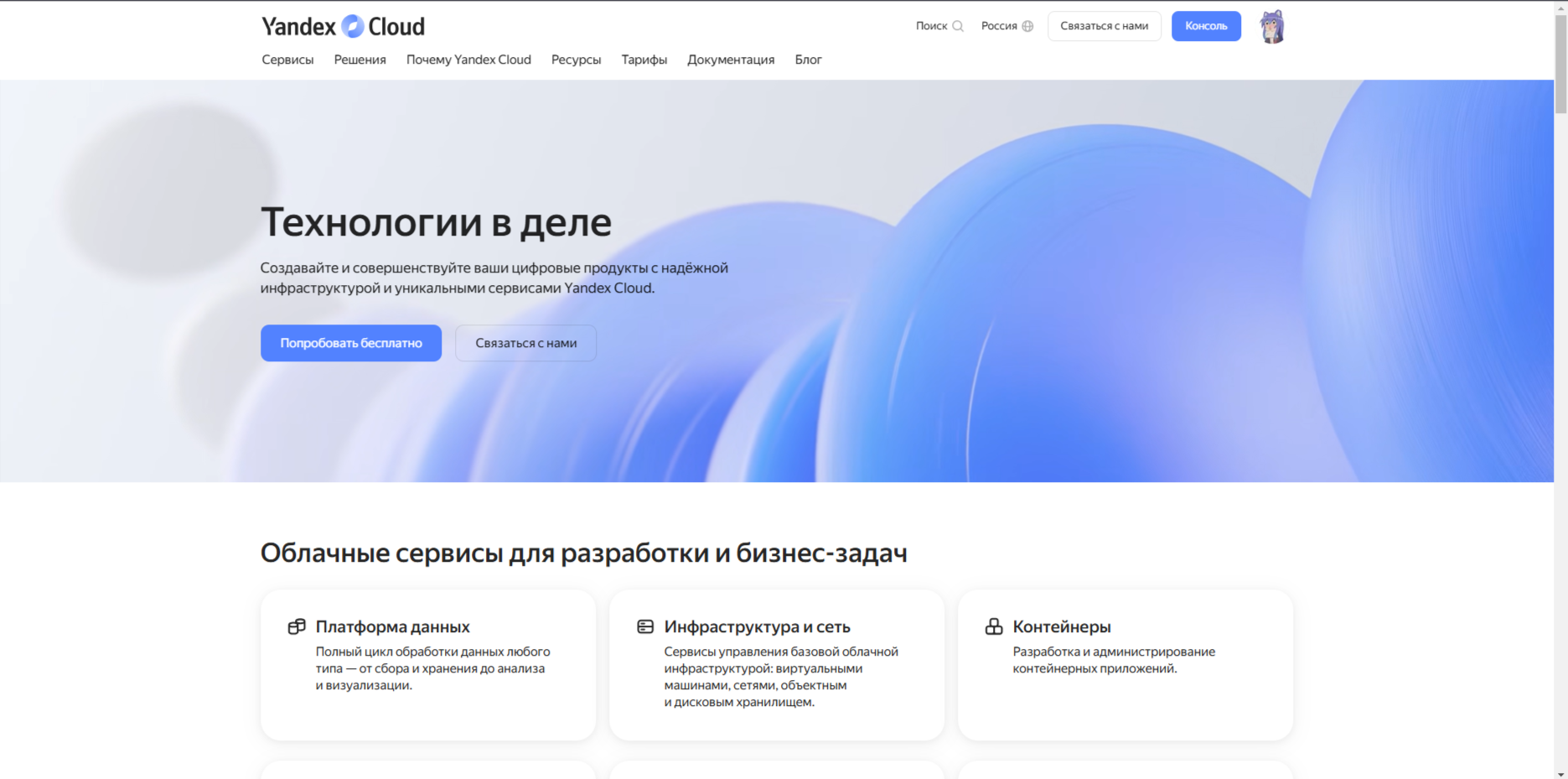
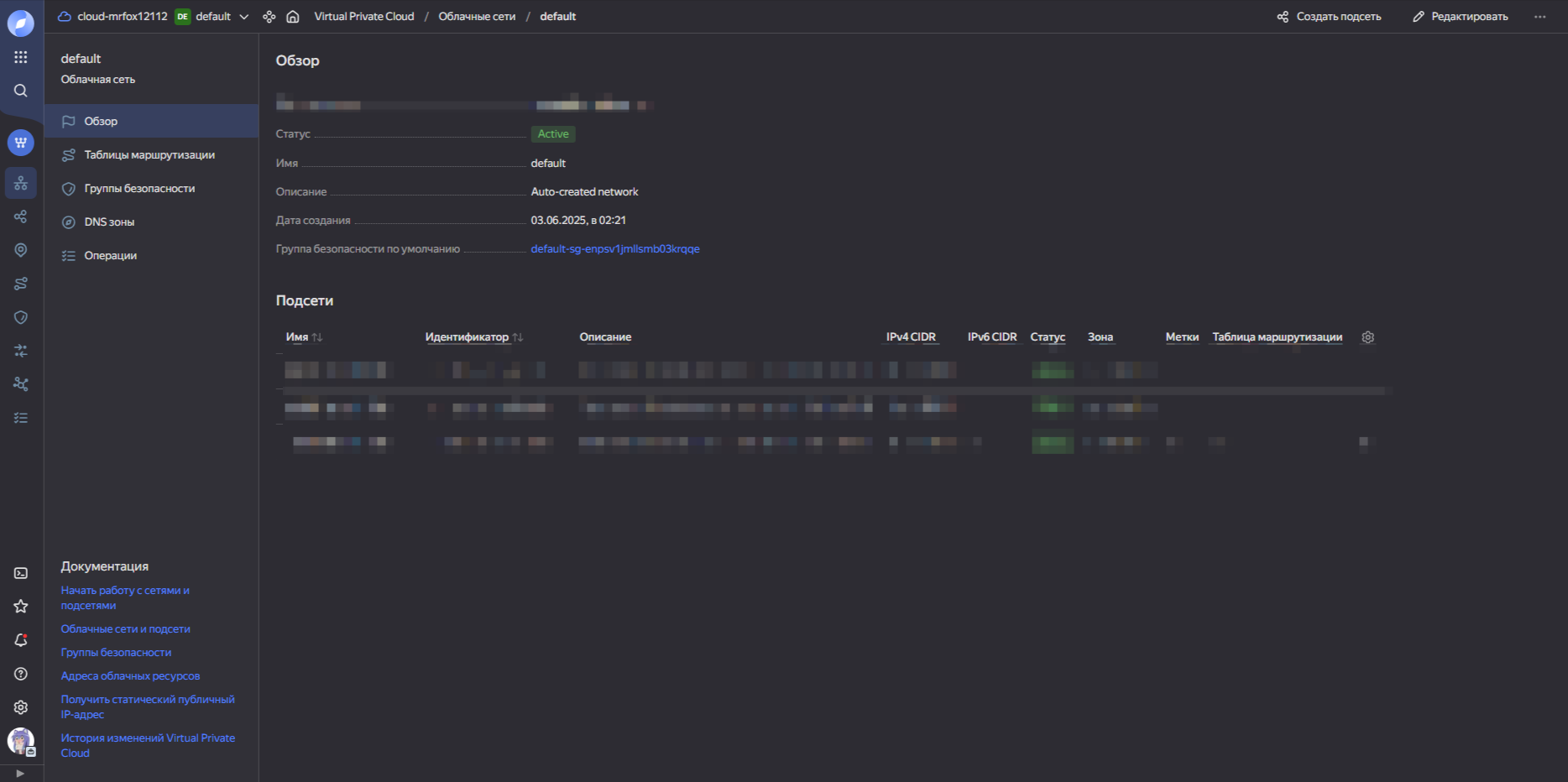


Рисунок 2 – Главная страница Dashboard Yndex Cloud

Рисунок 3 – Страница созданной приватной облачной сети



В свою очередь сервис Cloud (от Сбер) предоставляет собственные AI-разработки, также он имеет хорошую интеграцию с корпоративными решениями и имеет гибкие тарифы и скидки для предприятий, что в долгосрочных перспективах будет дешевле, чем у Yandex Cloud, но стоит отметить, что Cloud будет сложнее в освоении, так как имеет менее интуитивно понятный интерфейс, чем у его аналогов, также он имеет меньше готовых AI-API и не имеет поддержки таких фреймворков как TenserFlow и PyTorch.

Рисунок 4 – Главная страница Cloud

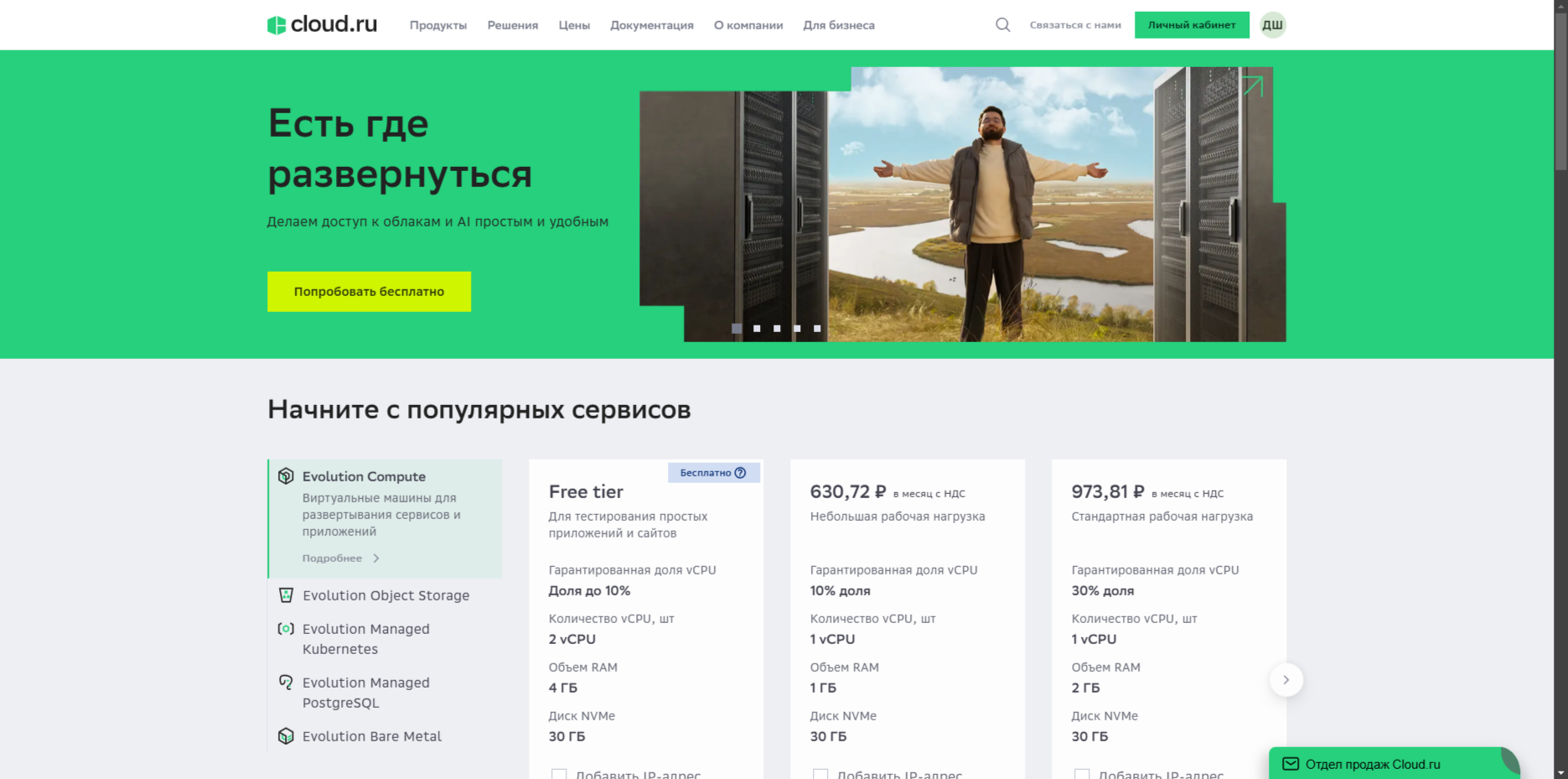


Рисунок 5 – Главная страница Dashboard Cloud

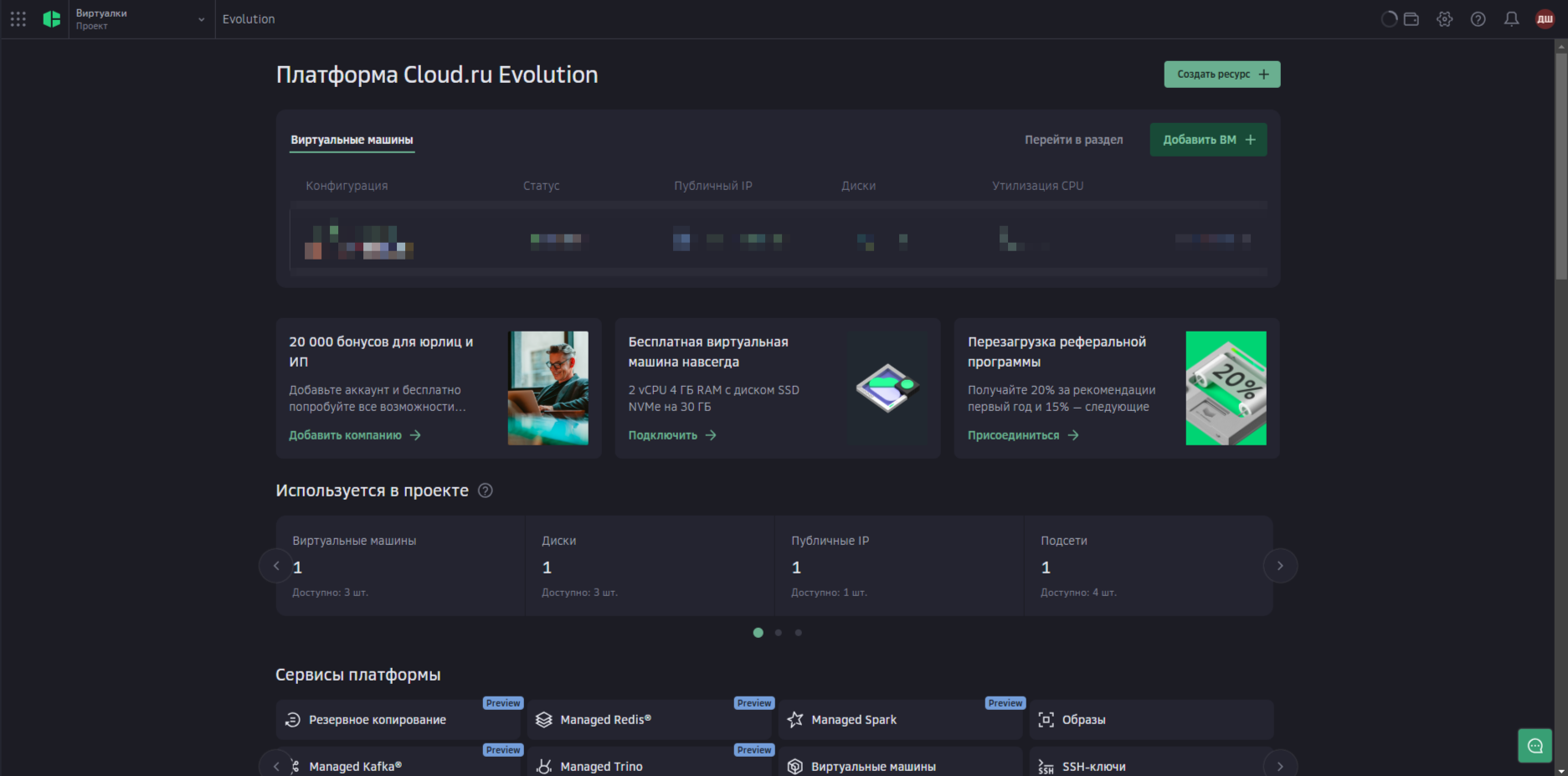
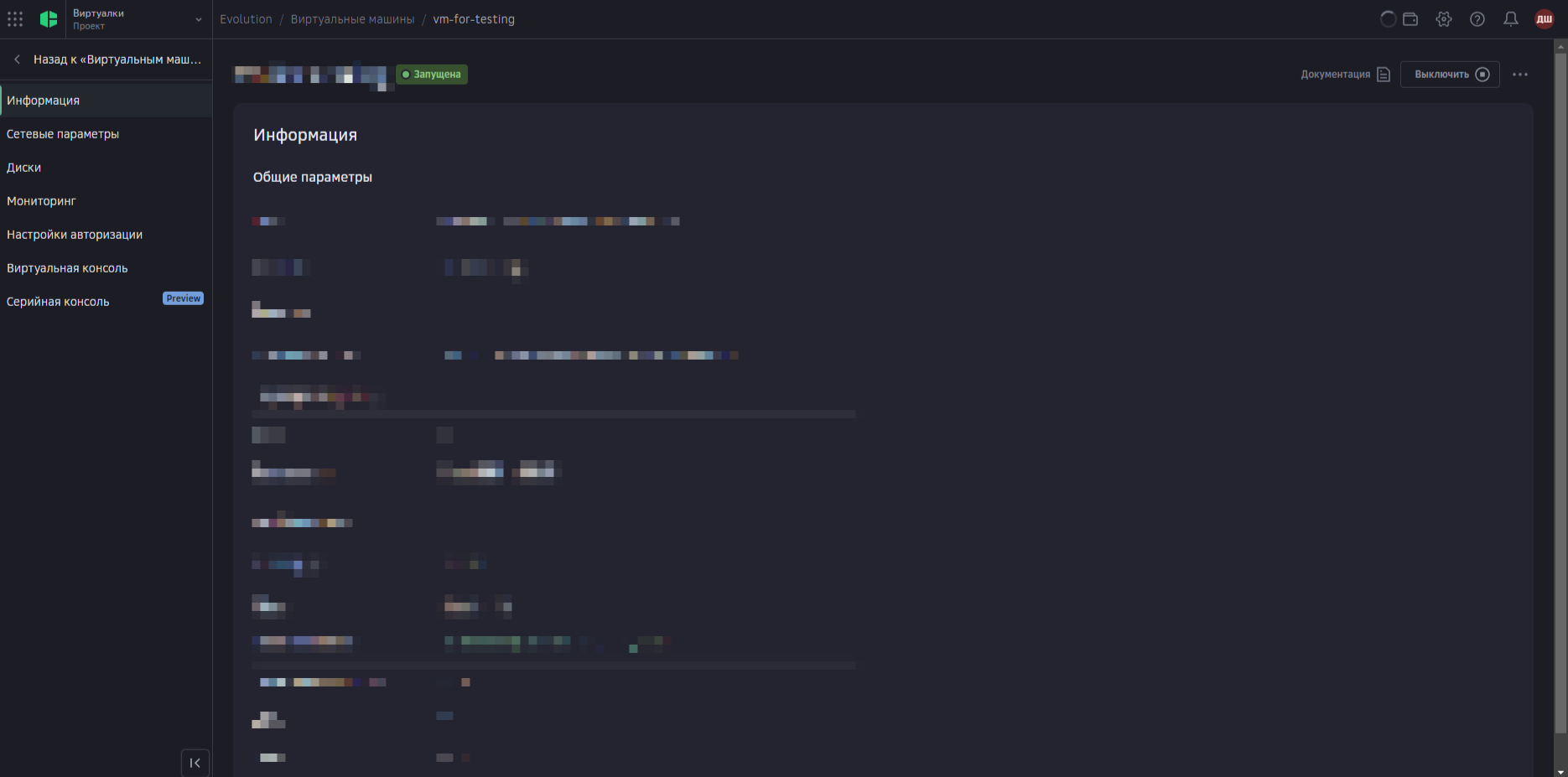


Рисунок 6 – Страница конкретного хостинга



# 2 Специальная опытно-экспериментальная часть

## 2.1 Анализ задачи

2.1.1 Информационное моделирование предметной области

При разработке структуры базы данных для модулей взаимодействия в СУБД PostgreSQL было спроектировано и реализовано 11 таблиц. Эти таблице охватывают ключевые сущности, необходимые для работы модулей взаимодействия, обеспечивая хранение и обработку данных.

Между таблицами были установлены отношения “один-к-одному”, “один-ко-многим” и “многие-к-одному”, что обеспечивает управление и целостность данных, схема данных предоставлена на рисунке А.5, также для более подробного изучения связей и таблицу была сделана ER диаграмма, предоставленная на рисунке А.4.

Основные таблицы с примерами:

Аккаунты пользователей – таблица содержит необходимые сведения о зарегистрированных пользователях.

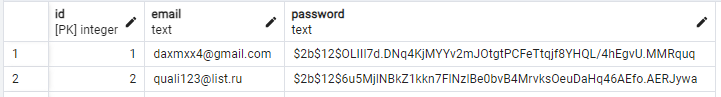


Рисунок 7- Данные таблицы аккаунтов

Рабочие пространства – данная таблица содержит всю необходимую информацию о рабочих пространствах.

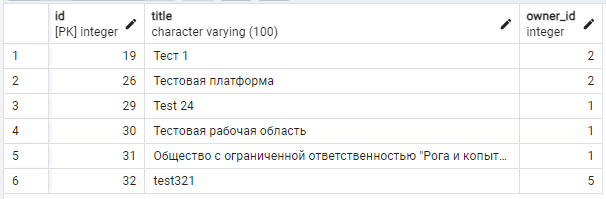


Рисунок 8 – Данные таблицы рабочих пространств

Внутренние пользователи системы – таблица содержит информацию об внутренних аккаунтах, находящихся в рабочих пространствах.

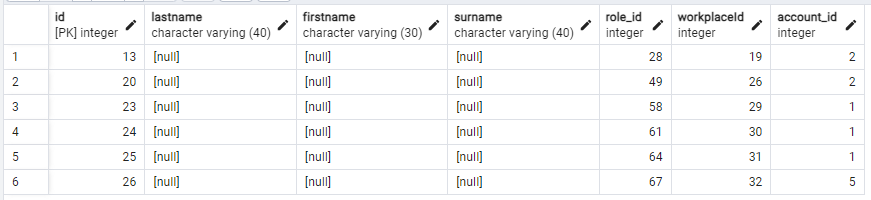


Рисунок 9 – Данные таблицы внутренних пользователей

Роли пользователей – таблица хранит в себе информацию о ролях пользователей в рабочих пространствах.

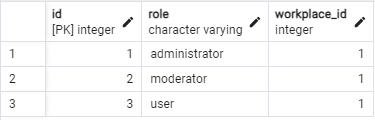


Рисунок 10 – Данные таблицы ролей

Коды подтверждения – таблица хранит коды подтверждения, для смены пароля от аккаунта.

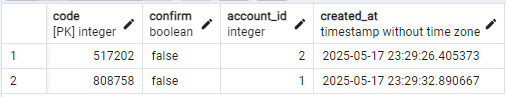


Рисунок 11 – Данные таблицы кодов подтверждения

Активные токены – данная таблица содержит список действующих токенов, где каждая запись имеет уникальный идентификатор токена (jti), дату и время создания, а также четкую связь с конкретным пользователем системы.

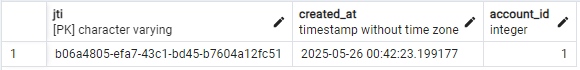


Рисунок 12 – Данные таблицы активных токенов

Хостинги – таблица содержит в себе информацию о всех хостингах.

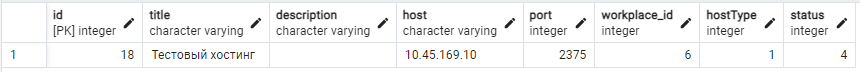


Рисунок 13 – Данные таблицы хостингов

Интеграции – данная таблица хранит в себе информацию о всех интеграциях, где каждая интеграция включает в себя уникальный идентификатор (uuid), название интеграции, её подробное описание, а также содержит связь с соответствующим рабочим пространством.

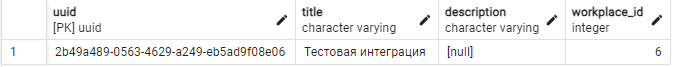


Рисунок 14 – Данные таблицы интеграций

Контейнеры интеграции – таблица содержит в себе информацию о контейнерах, привязанных к интеграции.

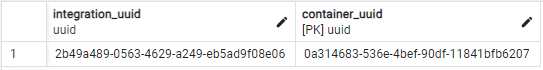


Рисунок 15 – Данные таблицы контейнеров интеграции

Права ролей – данная таблица хранит в себе информацию о всех доступных разрешениях и ограничениях, которые применяются к конкретным ролям в рамках модулей взаимодействия.

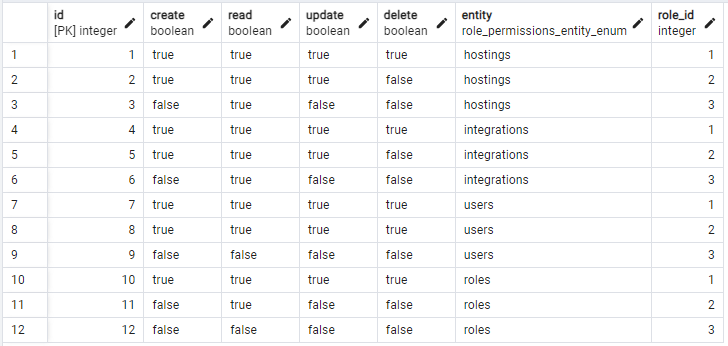


Рисунок 16 – Данные таблицы прав ролей

Права пользователей – данная таблица содержит подробную информацию о разрешениях и уровнях доступа, назначенных каждому конкретному пользователю в рамках работы с модулями взаимодействия.

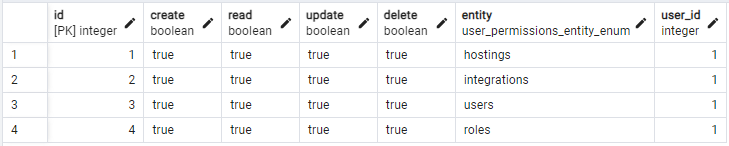


Рисунок 17 – Данные таблицы прав пользователей

2.1.2 Проектирование пользовательского интерфейса

Создание пользовательского интерфейса, это ключевой этап в разработке модулей взаимодействия, от того насколько удобным и понятным будет интерфейс, зависит насколько эффективно пользователи будут взаимодействовать с нейросетевыми моделями и интеграциями. В рамках дипломного проекта реализовано веб-приложение на базе vue.js [5] и серверное приложение на базе nest.js [6], система позволяет легко ориентироваться всем пользователям: администраторам, модераторам и пользователям.

При первом запуске веб-приложения перед пользователем открывается страница авторизации. Данная страница содержит специальную форму, через которую пользователь может зайти в свой аккаунт, введя необходимые данные. Кроме этого, на этой же странице есть возможность перейти на другие связанные страница – например, если у пользователя ещё нет аккаунта, он может перейти на страницу регистрации, также предусмотрена опция для тех, кто забыл пароль – на странице авторизации есть ссылка, ведущая на страницу восстановления пароля.

На странице авторизации расположены следующие элементы:

* форма авторизации, расположенная по центру;
* заголовок формы;
* поле ввода почты;
* поле ввода пароля;
* переключатель c надписью: “Запомнить меня”;
* ссылка на страницу восстановления пароля;
* кнопка входа в систему;
* кнопка перехода на страницу регистрации.

После авторизации пользователя, если у него есть рабочие пространства, его перенаправит на страницу выбора рабочих областей. Если у пользователя ещё нет аккаунта, то он может перейти на страницу регистрации и создать аккаунт.

На странице регистрации расположены элементы:

* форма регистрации, расположенная по центру;
* заголовок формы;
* поле ввода почты;
* поле ввода пароля;
* поле ввода для повторения пароля;
* кнопка регистрации в системе;
* кнопка перехода на страницу авторизации.

После регистрации пользователя в системе, его перенаправит на страницу создания рабочего пространства.

На странице создания рабочего пространства расположены элементы:

* форма создания рабочего пространства, расположенная по центру;
* заголовок формы;
* поле ввода названия рабочего пространства;
* кнопка назад, чтобы вернуться к выбору рабочих, доступна только если у пользователя есть рабочие пространства;
* кнопка для создания рабочего пространства.

Если у пользователя есть рабочее пространство, то после авторизации, его перенаправит на страницу выбора рабочих областей.

На странице выбора рабочих областей расположены элементы:

* форма выбора рабочего пространства, расположенная по центру;
* заголовок формы;
* список рабочих областей, с возможность выбора любой;
* кнопка, позволяющая перейти на страницу создания рабочего пространства.

После авторизации пользователя в системе и выбора или создания рабочего пространства, пользователя перенаправит на страницу со списком хостингов. Для пользователя после авторизации есть несколько страниц:

* страница хостингов – содержит список хостингов, с возможностью отфильтровать их по нужным параметрам, а также на странице присутствует кнопка, открывающая модальное окно с возможностью создать хостинг;
* страница конкретного хостинга – расположена информация по хостингу, с возможностью настроить его или просмотреть статистику работы;
* страница интеграций – расположен список интеграций, с фильтром и возможностью создать интеграцию;
* страница конкретной интеграции – содержится информация по конкретной интеграции, с возможностью настройки и добавления связки с контейнерами;
* страница настроек – расположена информация о рабочем пространстве, пользователях и ролях, также на странице можно настроить права ролей или пользователей, а также создать новую роль или пользователя;
* страница настроек аккаунта – выведены данные аккаунта и пользователя системы. Также на странице присутствует возможностью настроить пользователя системы или сменить пароль от аккаунта.

## 2.2 Описание логической структуры

2.2.1 Алгоритм программы

Модули взаимодействия разработаны с помощью vue.js и nest.js, основных на языке JavaScript с надстройкой в виде TypeScript, также в качестве базы данных используется PostgreSQL.

Функциональное назначение данных модулей заключается в оптимизации ключевых процессе: создание хостинга для взаимодействия с нейросетевыми моделями, создание и управление интеграциями.

Основные алгоритмические блоки программы:

* работа с хостингами:
  1. отображение хостингов – хостинги подгружаются из базы данных и выводятся в виде списка на странице хостингов, с возможность создания хостинга и их фильтрации;
  2. создание хостинга – пользователь может создать хостинг, выбрав нужную модель, указав название хостинга и описание (опционально), а также выбрать нужную интеграцию (если такая имеется);
  3. управление хостингом – при достаточных правах, пользователь может как удалить, так и поменять какую-либо информацию о хостинге;
  4. просмотр статистики хостинга – пользователь может просматривать статистику работы хостинга, а также метрики работы нейросети.
* работа с интеграциями:
  1. отображение интеграций – интеграции подгружаются из базы данных и выводятся в виде списка на странице интеграций, с возможностью создать интеграцию и отфильтровать их фильтрации;
  2. создание интеграции – пользователь может создать интеграцию, введя в форму название и описание, а также указав список контейнеров, к которым будет привязана создаваемая интеграция;
  3. управление интеграциями – пользователь с достаточными правами, может как удалить интеграцию, так и изменить данные самой интеграции.
* работа с рабочими пространствами:
  1. выбор рабочего пространства – если у пользователя уже есть рабочие пространства, он может выбрать одно из имеющихся и начать в нём работу;
  2. создание рабочего пространства – после авторизации, пользователь может создать своё рабочее пространство, где он сможет работать с хостингами, интеграциями и пользователями;
  3. управление рабочим пространством – пользователь, который является создателем системы, может обновить, а также удалить рабочее пространство.
* работа с пользователями:
  1. отображение пользователей – пользователи подгружаются из базы данных и выводятся в виде списка на странице настроек с выбранной вкладкой “пользователи”, также присутствует возможность найти нужного пользователя по почте;
  2. создание пользователя – модератор или администратор может пригласить пользователя в рабочее пространство по почте, приглашенному пользователю отправляется письмо с сообщением, что он добавлен в рабочее пространство, а также ему выдаётся логин, который будет являться его почтой и сгенерированный пароль;
  3. управление пользователями – происходит на странице настроек выбранного пользователя из списка, имеется возможность поменять роль, а также отрегулировать права на работу с хостингами и интеграциями.

2.2.2 Используемые методы

В процессе разработки модулей взаимодействия применялись как общепринятые, так и специализированные методы, обеспечивающие надёжность, производительность и эффективную обработку информации.

Методы работы с базой данных:

* нормализация данных – таблицы спроектированы в соответствии с нормальными формами, что исключает избыточность и непредсказуемость при обновлении данных;
* оптимизация производительности – были подобраны подходящие типы для хранения данных, а также для ускорения поиска необходимых данных, были установлены индексы на часто используемых полях.

Методы защиты данных:

* преобразование паролей – все пользовательские пароли подвергаются криптографическому преобразованию с помощью алгоритма BCrypt;
* контроль доступа – реализована многоуровневая система доступа, где, каждому пользователю присваивается определённая роль, для каждой роли определён определённый перечень разрешённых действий, также попытки неавторизованных операций автоматически блокируются;
* безопасное формирование запросов – все обращения к СУБД проходят через ORM, где каждый запрос обрабатывается и предотвращает какие-либо возможные инъекции.

Блок-схему создания рабочего пространства можно просмотреть на рисунке А.6. Блок-схему алгоритма создания хостинга можно просмотреть на рисунке А.7.

Также в качестве метода авторизации, стоит отметить генерацию токенов доступа к системе. В качестве метода для генерации токенов используется HMAC для генерации самих токенов и библиотека uuid, которая генерирует уникальные идентификаторы для токенов. Также чтобы определять, что пользователь активен в системе, токены сохраняются в базе данных и при смене пароля, все активные токены пользователя очищаются.

2.2.3 Составные части программы и связи между ними

Модули взаимодействия реализованы как совокупность модулей, соединённых в 1 систему, где каждый модуль, является самодостаточным и имеет минимальное кол-во связей с другими модулями. Можно выделить следующие модули:

* модуль авторизации – обеспечивает авторизацию пользователя в свой аккаунт, он реализует процессы входа в систему и проверку на наличие у пользователя рабочих пространств, а также выдаёт персональный токен доступа к системе;
* модуль регистрации – непосредственно регистрирует пользователя в системе, а также при успешной регистрации выдаёт ему токен доступа;
* модуль восстановления пароля – имеет 3 стадии, на которых пользователь сначала указывает почту, потом код пришедший на указанную почту, а на конечной стадии указывает новый пароль;
* модуль рабочего пространства – позволяет пользователю как создать рабочее пространство, так и выбрать из уже имеющихся, а также если пользователь является создателем, то он может как настроить, так и удалить рабочее пространство;
* модуль хостингов – позволяет пользователю в зависимости от прав взаимодействовать с хостингами;
* модуль интеграций – предоставляет возможность пользователю в зависимости от прав, взаимодействовать с интеграциями;
* модуль ролей – позволяет уполномоченному пользователю взаимодействовать с ролями, как настраивать и добавлять, так и удалять их, а также он может настраивать права ролей на разные сущности;
* модуль пользователей – позволяет пользователю с достаточными правами взаимодействовать с другими пользователями, как настраивать и добавлять, так и удалять их, а также он может настраивать права пользователей на разные сущности;
* модуль аккаунта – пользователь может взаимодействовать с общим для всех рабочих областей аккаунтом;
* Модуль текущего пользователя – пользователь может взаимодействовать со своим профиль пользователя для конкретной системы.

## 2.3 Описание работы программы

2.3.1 Общие сведения

Модули взаимодействия представляют собой комплекс модулей, разработанные для автоматизации ключевых бизнес-процессов: управление нейросетевыми моделями, мониторинг работы моделей, интеграция нейросетей в сервисы. Модули созданы на основе vue.js для пользовательского интерфейса и nest.js для серверной части, в качестве базы данных используется СУБД PostgreSQL. Все модули взаимодействуют друг с другом через единую базу данных, где хранятся все таблицы содержащие всю информацию об аккаунтах, рабочих областях, пользователей внутри рабочих областей, хостингах, интеграциях, ролях и правах.

2.3.2 Функциональное назначение

Модули взаимодействия обеспечат автоматизацию работы с нейросетями, что позволит минимизировать человеческий фактор в анализе и обработке данных. Модули захватывают следующие основные функциональные блоки:

* управление аккаунтом и пользователем – модуль позволяет пользователям настраивать свой общий аккаунт для всех рабочих областей и свой личный аккаунт в каждом рабочем пространстве;
* управление хостингами – модуль позволяет гибко управлять хостингами, нейросетевыми моделями и просматривать статистику работы хостинга и метрики модели;
* управление рабочим пространством – модуль позволяет создавать и настраивать роли, управлять правами пользователей и настраивать общую информацию о рабочем пространстве;
* управление интеграциями – пользователь может создать интеграцию со сторонним сервисом и привязать её к хостингу.

2.3.3 Связи с другими программами

Модули взаимодействуют с информационной системой управления нейросетями, всё основное взаимодействие происходит в модуле хостингов.

Взаимодействие также происходит и при интеграции с сервисами, например, с телеграммом: пользователь создаёт интеграцию в системе и получает уникальный ключ интеграции, после чего он создаёт своего телеграмм бота и настраивает обращения на открытие контроллеры интеграций у сервера модулей взаимодействия, так при каждом обращении к интеграции, сервер с помощью простого балансировщика нагрузки отправит запрос на связанный с интеграцией и минимально нагруженный контейнер.

В сами модули интегрировано взаимодействие с внутренней базой данных PostgreSQL. Такое решение позволяет внутри модулей хранить состояние данных получаемых от сторонних сервисов и интеграций.

2.3.4 Входные данные

Модули взаимодействия предназначены для работы с различными типами входящих данных. Они способны принимать и обрабатывать информацию, относящуюся к нескольким категориям.

Основной системы являются данные пользователей, включающие почту и пароль. Эти параметры позволяют идентифицировать пользователя в систему, а также генерировать токен доступа к защищённым данным.

Вторым, но не менее важным блоком входных данных будут являться данные рабочего пространства. В частности, речь идёт о названии рабочего пространства, списке внутренних пользователей, ролей, а также сведенья о создателе данного рабочего пространства.

Третьим блоком входных данных можно отметить данных внутреннего пользователя рабочего пространства. Здесь существует целый комплекс персональных и служебных данных: фамилия, имя, отчество, его роль в рабочем пространстве, а также привязанный аккаунт и рабочее пространство, к которому пользователь привязан.

Четвёртым блоком стоит ответить данные интеграций. Эти данные включают в себя название и описание интеграции, а также список контейнеров в которых развёрнуты нейросети.

2.3.5 Выходные данные

Модули взаимодействия направлены на формирование выходных данных, которые играют ключевую роль в управлении нейросетями.

Одним из важнейших стоит отметить статистику работы нейросети, которая включает в себя информацию о хостинге и о нейросетевых моделях за определённый период времени.

Ещё одним важным типом выходных данных нужно отметить идентификаторы интеграций, которые нужны для отправки данных на открытый контроллер интеграций.

2.3.6 Тестирование

В рамках дипломного проекта было проведено тестирование модулей взаимодействие несколькими методами:

* метод ручного тестирования – были разработаны тест-кейсы, которые включают в себя ряд шагов, тестируемых данных и ожидаемые результаты, тестирование охватило основные функции программы, такие как авторизация пользователя, создание хостинга, создание интеграции;
* метод unit-тестирования – тестирования основного кодового функционала системы, эти тесты выполняются при вызове определённой команды и если тестирование не прошло, то вносятся своевременные изменения.

2.3.7 Вызов и загрузка

Модули взаимодействия размещены на удалённом сервере и доступны непосредственно через веб-браузер. Это означает, что пользователям не требуется устанавливать приложение, достаточно просто перейти по соответствующей ссылке.

При первом открытии веб-приложения, пользователя автоматически перенаправляет на страницу авторизации, где необходимо войти в свой аккаунт или зарегистрироваться.

Такая организация работы обеспечивает максимально быстрый старт, всё что требуется это браузер и доступ в интернет.

## 2.4 Руководство оператора

2.4.1 Назначение программы

Модули взаимодействия разработаны для автоматизации ключевых бизнес-процессов, включая управление нейросетевыми моделями, анализ больших данных и интеграция нейросетей в сторонние сервисы.

Целевой аудиторий модулей будут являться следующие категории пользователей:

* бизнес-аналитики – могут анализировать большие данных, прогнозировать тренды и оптимизировать процессы;
* разработчики – смогут создавать, настраивать и управлять нейросетевыми моделями, а также интегрировать их в различные сервисы.

Модули взаимодействия служат хорошим инструментом, для управления нейросетями и их интеграций в различные сервисы, что позволит снизить нагрузку и оптимизировать необходимые процессы.

2.4.2 Условия выполнения

Для функционирования модулей взаимодействие, нужен сервер, который должен соответствовать минимальным требованиям:

* операционная система Ubuntu 20.4 или выше;
* процессор не ниже Intel Core i3;
* оперативная память не меньше 4ГБ;
* место на жёстком диске не меньше 30 ГБ.

Для успешного функционирования модулей, на сервере должны быть предустановлены node.js и PostgreSQL.

2.4.3 Выполнение программы

2.4.3.1 Работа пользователя

После перехода пользователя на веб-приложение, будет открыта страница авторизации. Если у пользователя есть аккаунт, он может зайти в него и выбрать рабочее пространство, где он является простым пользователем. Данные страницы отображены на рисунках с Б.1 до Б.5.

Основные функции доступные пользователю:

* станица хостингов – пользователь может просмотреть существующие хостинги, также может перейти к какому-то конкретному хостингу;
* страница конкретного хостинга – на странице пользователь увидит настроенные данных, без возможности их редактирования, а также сможет просмотреть статистику работы самого хостинга;
* страница интеграций – на самой странице, пользователь сможет увидеть список интеграций, с возможностью перейти к конкретной интеграции;
* страница конкретной интеграции – на данной странице пользователь увидит общую информацию по интеграции, но без возможности редактирования и просмотра данных для связки;
* страница настройки аккаунта – на странице пользователь сможет просмотреть и настроить данные от своего общего аккаунта и от пользователя, привязанного к аккаунту.

2.4.3.2 Работа модератора

Для входа, у модератора уже должен быть аккаунт, и он должен быть добавлен в рабочее пространство с установленной ролью “модератор”.

После входа ему будет доступен следующий функционал:

* управление пользователями:
  1. добавлять или удалять пользователей;
  2. редактирование информации пользователей.
* управление хостингами:
  1. добавление хостинга;
  2. редактирование хостинга.
* управление интеграцией:
  1. добавление интеграции;
  2. редактирование интеграции.

Это позволит выполнять модераторам минимальный функционал управления, без привлечения администраторов системы.

2.4.3.3 Работа администратора

Для входа, у администратора должен уже быть аккаунт, и он должен либо являться создателем рабочего пространства или должен быть добавлен в рабочее пространство с ролью “администратор”.

После входа в систему, ему будет доступен следующий функционал:

* управление рабочим пространством:
  1. возможность изменить общую информацию о рабочем пространстве;
  2. управление ролями и доступами к различной функциональной части модулей взаимодействия.
* управление хостингами:
  1. возможность удалить хостинг.
* управление интеграциями:
  1. возможность удалить интеграцию.

Администратору, доступно чуть больше функционала, чем модератору, но данный функционал, может быть ключевым при командной работе с модулями взаимодействия.

# 3 Экономическая часть

## 3.1 Расчёт заработной платы с отчислением на социальное страхование

3.1.1 Определение перечня работ и трудоёмкости разработки программного продукта

Смета затрат на производство представляет собой общий свод плановых расходов в стоимостном выражении, необходимых для выпуска продукции, выполнения работ и оказания услуг в рамках производственной программы предприятия. Разработка сметы осуществляется по единой номенклатуре экономических элементов.

Структура затрат организации зависит от особенностей отрасли и определяется типовыми рекомендациями по учёту и калькулированию себестоимости, разработанными министерствами и ведомствами при согласовании с Минфином РФ. Эти рекомендации применяются в промышленности, сельском хозяйстве, строительстве, торговле, общественном питании и других сферах деятельности.

Затраты включают:

* расходы, связанные с выпуском продукции, выполнением работ и оказанием услуг;
* расходы на управление и реализацию продукции, за исключением капитальных вложений, средств, направленных на освоение новых видов продукции, а также непроизводственных расходов.
* группировка затрат в смете проводится по следующим экономическим элементам:
* материальные затраты, включающие сырьё и материалы, покупные комплектующие, полуфабрикаты, топливо, запасные части и прочие аналогичные расходы;
* оплата труда, которая состоит из заработной платы, премий за достижение показателей, выплат за выслугу лет, резервов на оплату отпусков;
* отчисления в социальные фонды;
* амортизационные отчисления, относящиеся к основным фондам, нематериальным активам, малоценным и быстроизнашивающимся предметам;
* прочие расходы, такие как оплата услуг сторонних организаций, хозяйственные издержки, затраты на рекламу и сбыт продукции.

Расчёт сметы выполняется на основе данных бизнес-плана, связанного с производственной программой на будущий финансовый год, анализа фактических показателей предыдущего периода и действующих нормативов по статьям расходов, принятых в организации.

Нормативы по статьям затрат устанавливаются с учётом рыночных цен на продукцию прошедший период. По этим данным рассчитывается максимально допустимый уровень издержек на производство, соответствующий текущему уровню рыночной или договорной цены. Полученные значения позволяют спрогнозировать объём затрат на единицу продукции и составить плановую смету с детализацией по статьям и элементам.

С учётом специфики отраслевой деятельности смета затрат формируется по направлениям работы, видам продукции, услуг или категориям деятельности. Дополнительно возможна группировка по структурным подразделениям.

На завершающем этапе планирования составляется сводная смета, ориентированная на производственную программу и объём выпуска продукции.

На начальном этапе создания программного продукта необходимо определить этапы разработки и оценить трудоёмкость каждого из них. В таблице 1 представлены все основные показатели.

Таблица 1 – Этапы и трудоёмкость выполнения работ В часах

|  |  |  |
| --- | --- | --- |
| Этапы разработки | Руководитель | Техник |
| Постановка задачи | 1 | 7 |
| Выбор инструментальных средств | 1 | 4 |
| Проектирование | 1 | 10 |
| Построение алгоритма | 1 | 20 |

Окончание таблицы 1 В часах

|  |  |  |
| --- | --- | --- |
| Этапы разработки | Руководитель | Техник |
| Программирование | 1 | 40 |
| Тестирование | 2 | 30 |
| Отладка | 1 | 13 |
| Написание пояснительной записки | 2 | 14 |
| Итого | 10 | 125 |
| 135 | |

Минимальный размер заработной платы в соответствии с установленными нормами составляет 22440 рубля.

Уровень заработной платы специалистов в сфере информационных технологий определён исходя из анализа текущего состояния рынка труда в Новгородской области. Были рассмотрены данные по вакансиям и предложениям о работе, публикуемые в открытых источниках, включая региональные сайты [4] по трудоустройству и отчёты служб занятости населения [3].

Месячный оклад руководителя проекта принят равным 34 400 руб., техника – 30 000 руб.

3.1.2 Расчёт основной заработной платы

При расчёте основной заработной платы учитывается заработная плата всех категорий работников, непосредственно занятых разработкой программного продукта. Размер заработной платы определяется исходя из количества исполнителей и их квалифицированного уровня, а также затраченного ими времени в целом на разработку программного продукта и отдельных его этапов.

Основная заработная плата работников определяется по повременной системе оплаты труда.

Рассчитаем цену одного часа работы сотрудника по формуле:

|  |  |
| --- | --- |
| Цч = Ом/(Др×Ч), | (1) |

где Др – среднее число рабочих дней в месяце;

Ч – часов в смене.

В соответствии с установленными окладами и нормативами рабочего времени, для расчёта часовой ставки используется число рабочих дней в мае 2025 года – 18 дней, согласно производственному календарю Российской Федерации на 2025 год, утверждённому Министерством труда.

Продолжительность рабочей смены принята равной 8 часам, что соответствует стандартной 40-часовой рабочей неделе. Таким образом, произведем расчёт оплаты труда за час.

Руководитель проекта:

Цч = 34400/(18×8) = 238,89 руб.

Техник:

Цч = 30000/(18×8) = 208,33 руб.

Заработная плата работников по окладу рассчитывается по формуле:

|  |  |
| --- | --- |
| ЗПосн = Цч×Ч, | (2) |

где Ч – количество часов, затраченных на разработку программного продукта.

Таблица 2 – Основная заработная плата работников В рублях

|  |  |  |  |
| --- | --- | --- | --- |
| Должность | Количество часов, затраченных на разработку продукта, час | Оплата труда за 1 час | Заработная плата |
| Руководитель проекта | 10 | 238,89 | 2388,9 |
| Техник | 125 | 208,33 | 26041,25 |
| Итого | | | 28430,15 |

Таким образом, по показателям, представленным в таблице 2, основная заработная плата работников составила 28430,15 руб.

3.1.3 Расчет дополнительной заработной платы

Дополнительная заработная плата включает различные виды доплат сверх основной заработной платы: премии, доплату за работу в сверхурочное время, надбавки за профессиональное мастерство, оплату очередного и учебных отпусков и прочие виды доплат.

Дополнительная заработная плата устанавливается на предприятии в процентах от суммы основной заработной платы и составляет от 16 до 20%.

Таким образом, дополнительная заработная плата будет определяться по следующей формуле:

|  |  |
| --- | --- |
| ЗПдоп = ЗПосн×20%/100%, | (3) |

где ЗПосн – основная заработная плата, руб.

Значение ЗПосн возьмём из таблицы 2, и рассчитаем дополнительную заработную плату.

|  |
| --- |
| ЗПдоп = 28430,15×0,2 = 5686,03 руб. |

На основании вышеприведенных расчетов, делаем вывод, что основная и дополнительно заработная плата работников по созданию модулей взаимодействия для информационной системы управления нейросетями составит 68745,6+13749,12 = 34116,18 руб.

3.1.4 Расчёт отчислений на социальное страхование

Отчисления на социальное страхование составляют 30% от суммы основной и дополнительной заработной платы. Отчисления на социальное страхование рассчитываются по формуле:

|  |  |
| --- | --- |
| Зстр = (ЗПосн+ЗПдоп)×30%/100%, | (4) |

где ЗПосн – основная заработная плата, руб.;

ЗПдоп – дополнительная заработная плата, руб.

Значение ЗПосн возьмём из таблицы 2, а ЗПдоп рассчитывается по формуле 3. Вычислим величину отчислений с помощью формулы 4:

|  |
| --- |
| Зстр = 34116,18×30%/100% = 10234,85 руб. |

Таким образом, из расчетов видно, что отчисления на социальное страхование составят 10234,85 руб.

3.2 Расчёт стоимости материалов и лицензионного обеспечения

При создании программного продукта используется лицензионное программное обеспечение, поэтому необходимо включить стоимость лицензионных программ, используемых при разработке программного продукта, в смету затрат.

Сначала определим перечень лицензионных программ, срок их действия и норму установки в таблице 3.

Таблица 3 – Перечень программного обеспечения для выполнения работ

|  |  |  |
| --- | --- | --- |
| Наименование лицензионных программ | Норма установки, шт. | Цена, руб. |
| Microsoft Windows 11 Pro | 1 | 7999 |
| Microsoft Office 365 | 1 | 1495 |
| WebShotrm | 1 | 2278 |
| Итого | | 11772 |

Стоимость лицензионного программного обеспечения для разработки программного продукта составит 11772 руб.

Расчёт стоимости необходимых материалов для разработки программного продукта рассчитывается на основе норм расхода материальных ресурсов и оптовых цен на их приобретение по формуле:

|  |  |
| --- | --- |
| См = Нр×Ц, | (5) |

где Нр – норма расхода материальных ресурсов в натуральных единицах;

Ц – цена приобретения за единицу, руб.

Расчёт стоимости материалов представлен в таблице 4.

Таблица 4 – Расчёт стоимости материалов В рублях

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование материала | Норма расхода | Цена за единицу | Сумма |
| Флеш-накопитель USB 32Гб, шт. | 1 | 600 | 600 |
| Бумага, уп. | 0,3 | 347 | 104,1 |
| Картридж, шт. | 1 | 1500 | 1500 |
| Папка, шт. | 1 | 215 | 215 |
| Итого | | | 2419,1 |

Общая стоимость материалов составит 2419,1 руб.

Для расчета затрат на электроэнергию учитывается мощность используемого оборудования и время работы над проектом. В данном случае использовался ноутбук MSI Katana 17 B12U с мощностью 53.5 Вт, что эквивалентно 0,0535 кВт. Общее время работы над проектом для техника составило 125 часов. Потребление электроэнергии за это время рассчитывается как произведение мощности устройства на время работы: 0,0535×125 = 6,6875 кВт/ч.

Для городских домов с электроплитами, одноставочный тариф второго диапазона равен 4 руб. 21 коп. ,затраты на электроэнергию составят: 6,6875×4,21 = 28,15 руб.

Кроме того, для обеспечения стабильного подключения к сети использовался безлимитный тариф "Оптимально" от компании – Новотелеком, стоимость которого составляет 480 рублей в месяц. Поскольку работа над проектом заняла 125 часов, а среднее количество рабочих часов в месяце составляет 160 часов, доля времени работы над проектом от общего рабочего времени в месяце составила: 125/160 = 0,78125. Таким образом, затраты на интернет за время работы над проектом рассчитываются как: 480×0,78125 = 375 руб.

Общие затраты на электроэнергию и интернет за время работы над проектом составят: 28,15+375 = 403,15 руб.

Итого расходов (прямых) – 34116,18+10234,85+11772+2419,1+403,15 = 58945,28 руб.

3.3 Расчёт накладных расходов

Накладные расходы представляют собой дополнительные к основным расходам затраты на управление, организацию и обслуживание производства.

Не связаны напрямую с основным производством товаров или предоставлением услуг, не входят в стоимость материалов и оплату труда.

Накладные расходы закладываются в себестоимость товара, издержки его производства и обращения, но не прямо, а косвенно, пропорционально стоимости материалов и сырья, сумме заработной платы и так далее.

Накладные расходы составляют 10% от расходов на эксплуатацию и рассчитываются по формуле:

|  |  |
| --- | --- |
| Нр = (Рэксп×10%)/100%, | (6) |

где Рэксп – величина расходов на эксплуатацию, руб.

|  |
| --- |
| Нр = (58945,28×10%)/100% = 5894,53 руб. |

На основании выше рассчитанных данных составим смету затрат на программный продукт в таблице 5.

Таблица 5 – Смета затрат на программный продукт

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Сумма, руб. | Структура затрат, % |
| Основная и дополнительная заработная плата | 34116,18 | 52,62 |
| Отчисления на социальное страхование | 10234,85 | 15,78 |
| Стоимость материалов | 2419,1 | 3,73 |
| Стоимость лицензионного обеспечения | 11772 | 18,16 |
| Стоимость электроэнергии и интернета | 403,15 | 0,62 |
| Итог расходы на эксплуатацию | 58945,28 | 90,91 |
| Накладные расходы | 5894,53 | 9,09 |
| Итого | 64839,81 | 100 |

Таким образом, общая сумма затрат на разработку модулей взаимодействия с информационной системой управления нейросетями составит 64839,81 руб.

По результатам расчета построим диаграмму структуры затрат на программный продукт и представим её на рисунке 18.

Рисунок 18 – Структура затрат на программный продукт

3.4 Составление и расчёт цены реализации программного продукта

Цена является одним из самых важных экономических показателей. Её основная функция состоит в обеспечении выручки от реализации программного продукта, поэтому цена определяет прибыль и финансовую стабильность предприятия, его жизнеспособность.

Так как целью предпринимательской деятельности является получение прибыли, то цена реализации программного продукта будет представлять собой сумму затрат на его разработку и запланированную величину прибыли. Примем уровень рентабельности Р = 30%. Величину общих затрат на разработку программного продукта возьмём из таблицы 5, а стоимости материалов из таблицы 4.

Расчёт оптовой цены производится по формуле:

|  |  |
| --- | --- |
| Цопт = З×(1+Р), | (7) |

где З – общие затраты на разработку программного продукта, руб.;

Р – уровень рентабельности проекта, коэффициент.

Произведём расчет оптовой цены используя формулу 9:

|  |  |
| --- | --- |
| Цопт = 64839,81×(1+0,30) = 84291,75 руб. |  |

Расчёт прибыли от продажи продукта производится по формуле:

|  |  |
| --- | --- |
| П = Цопт - З, | (10) |
| П = 84291,753-64839,81= 19451,94 руб. |  |

Таким образом, можно сделать вывод, что данный продукт прибыльный и может быть реализован. Прибыль от продажи программного продукта составит 19451 руб. 94 коп.

# Заключение

Разработанные модули взаимодействия с информационной системой управления нейросетями являются современным и технологическим решением, отвечающее актуальным требованиях и потребностям различных пользователей.

Модули, разработаны на базе современных веб-технологий, таких как Vue.js, Nest.js и PostgreSQL. Каждая технология гарантирует высокую производительность и надёжность системы. Благодаря своей реактивной архитектуре Vue.js позволяет динамически отображать контент и обновлять его без перезагрузки страницы. Также благодаря тому, что Nest.js поддерживает асинхронное программирование, программа не будет ждать выполнение запроса и будет обрабатывать все запросы параллельно.

Разработанные модули взаимодействия были тщательно продуманы и протестированы: от авторизации и регистрации аккаунта до создания хостинга с нейросетью.

Таким образом, модули взаимодействия, отвечают актуальным потребностям, решая такие проблемы как взаимодействие с нейросетями, анализ данных и интеграция нейросетей в различные сервисы.

# Список литературы

1. ГОСТ 2.105–2019 Единая система конструкторской документации. Общие требования к текстовым документам - Москва: Стандартинформ, 2019. - 36 с.;
2. ГОСТ 19.701–2010 Единая система программной документации схемы алгоритмов, программ, данных и систем обозначения условные и правила выполнения - Москва: Стандартинформ, 2010. - 24 с.;
3. Департамент труда и занятости Новгородской области. [<https://mintrud.novreg.ru>] (Дата обращения: 27.05.25);
4. Работа в России. Федеральная служба по труду и занятости. [<https://trudvsem.ru>] (Дата обращения: 27.05.2025);
5. Vue.js Documentation [<https://vuejs.org/guide/introduction.html>] (Дата обращения 02.04.2025);
6. NestJS Documentation [https://docs.nestjs.com/] (Дата обращения 02.03.2025);
7. pgAdmin Development Team. pgAdmin 4: руководство по администрированию PostgreSQL. [<https://www.pgadmin.org/docs/>] (Дата обращения: 23.01.2025).

# Приложение А (Обязательно)

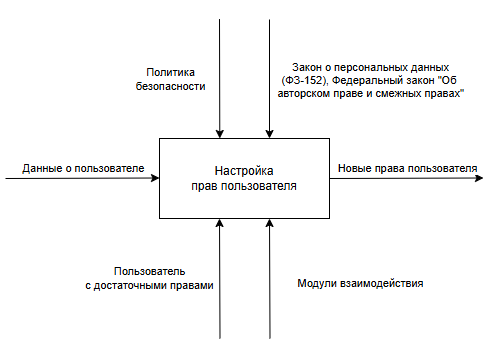


Рисунок А.1 – IDEF0 модели настройки прав пользователя

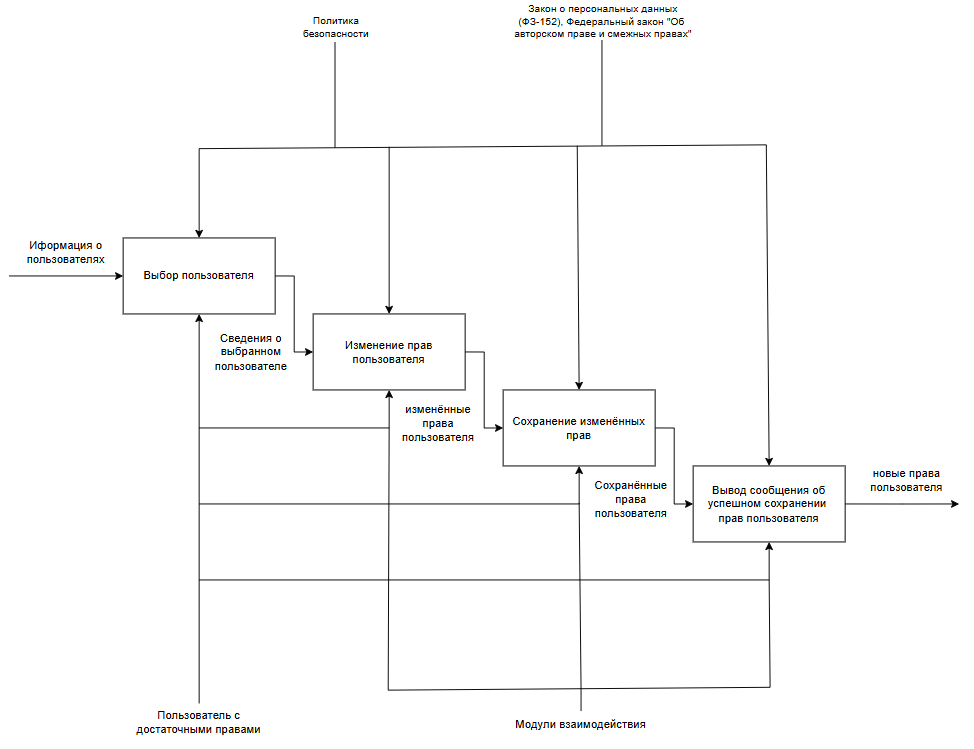


Рисунок А.2 – Декомпозиция IDEF0 модели настройки прав пользователя



Рисунок А.3 – Диаграмма прецедентов

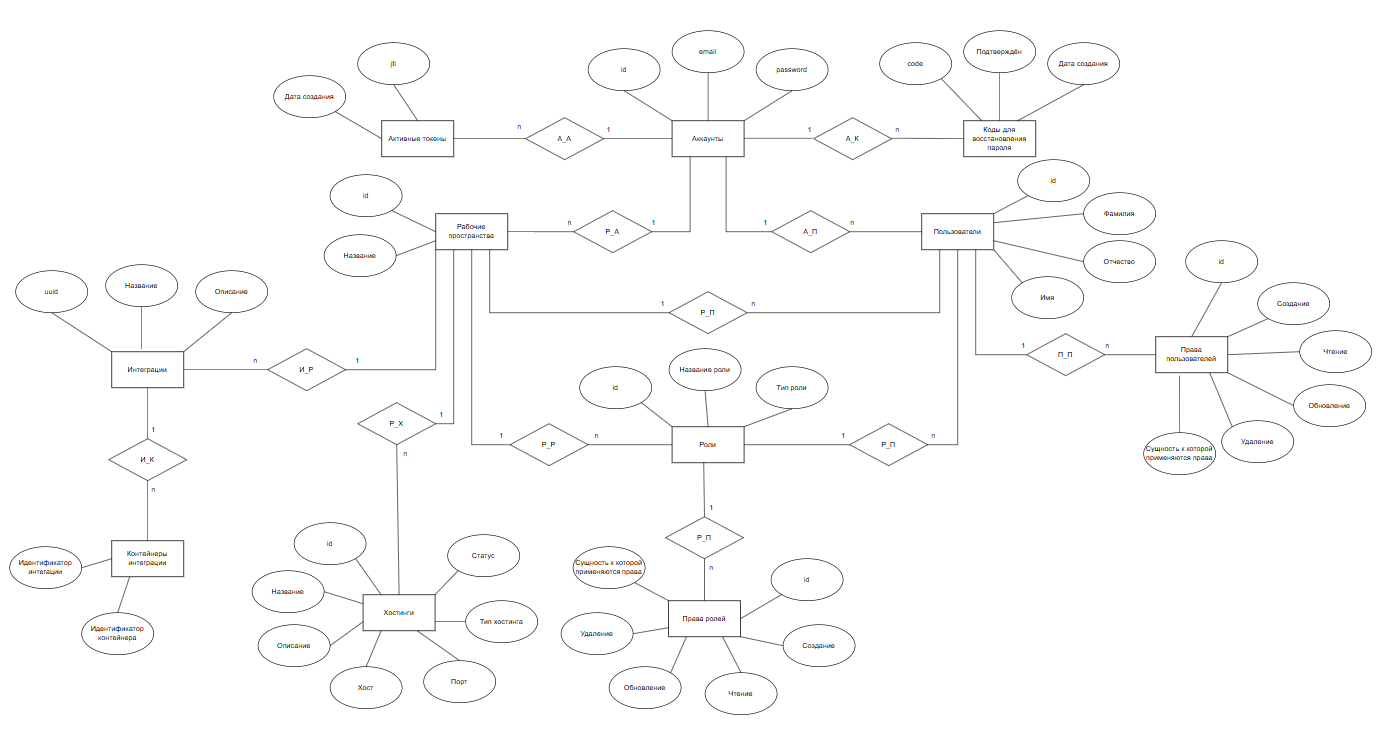


Рисунок А.4 – ER модель базы данных

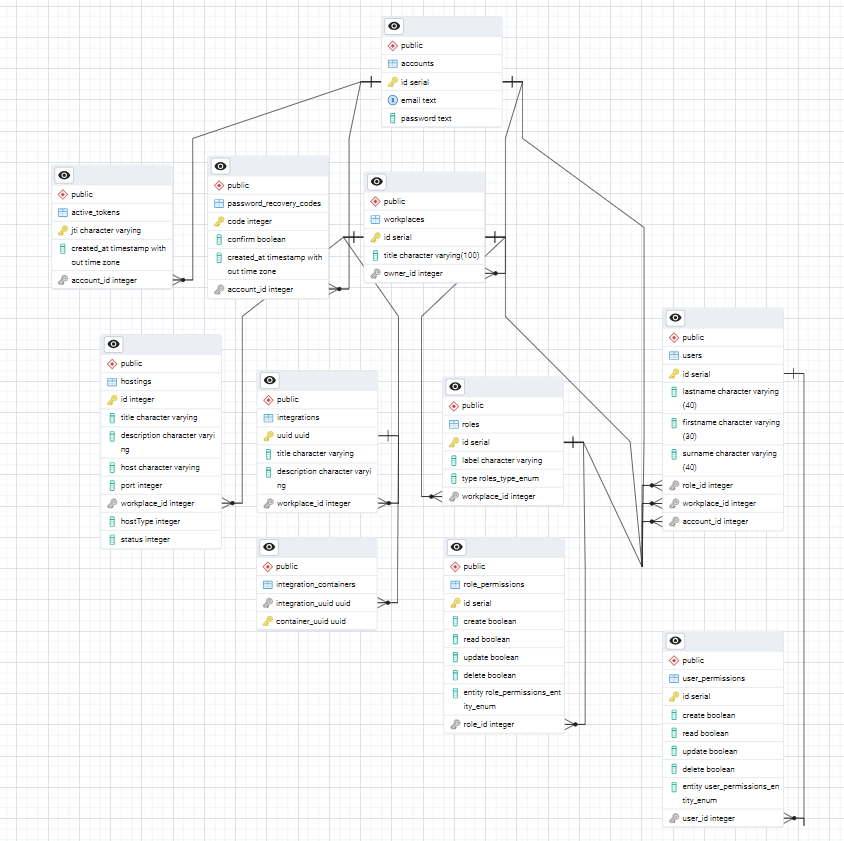


Рисунок А.5 – Схема данных

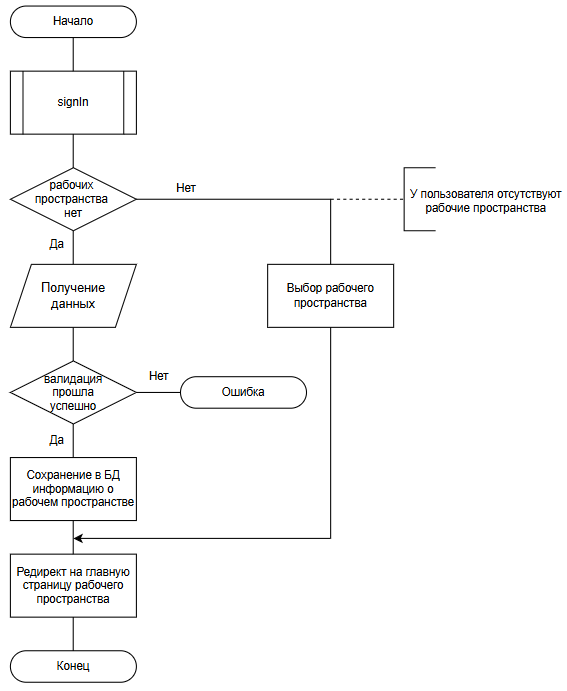


Рисунок А.6 - Блок-схема алгоритма создания рабочего пространства

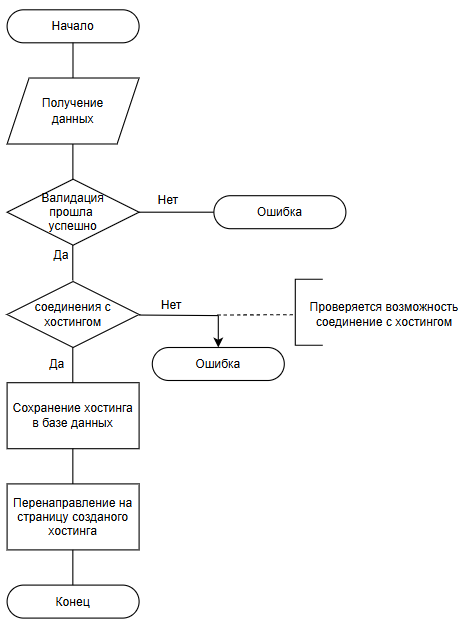


Рисунок А.7 - Блок-схема алгоритма создание хостинга

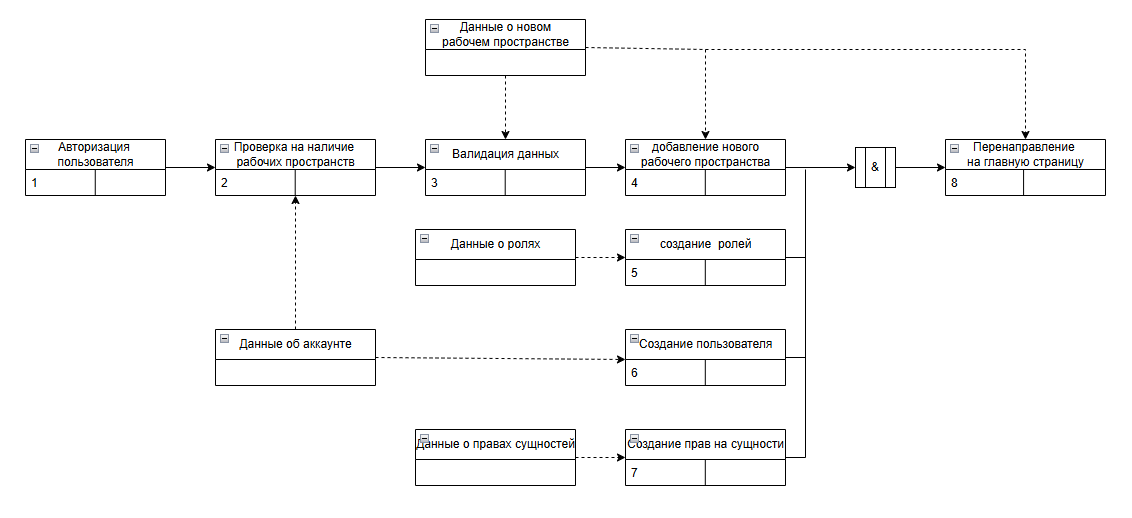


Рисунок А.8 - Модель IDEF3 создания рабочего пространства

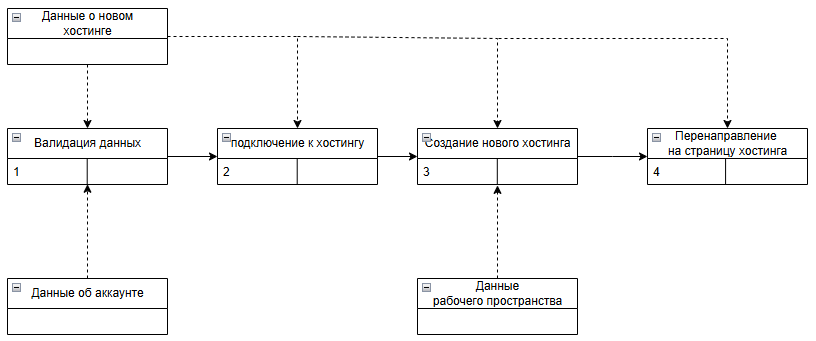


Рисунок А.9 - Модель IDEF3 создание хостинга

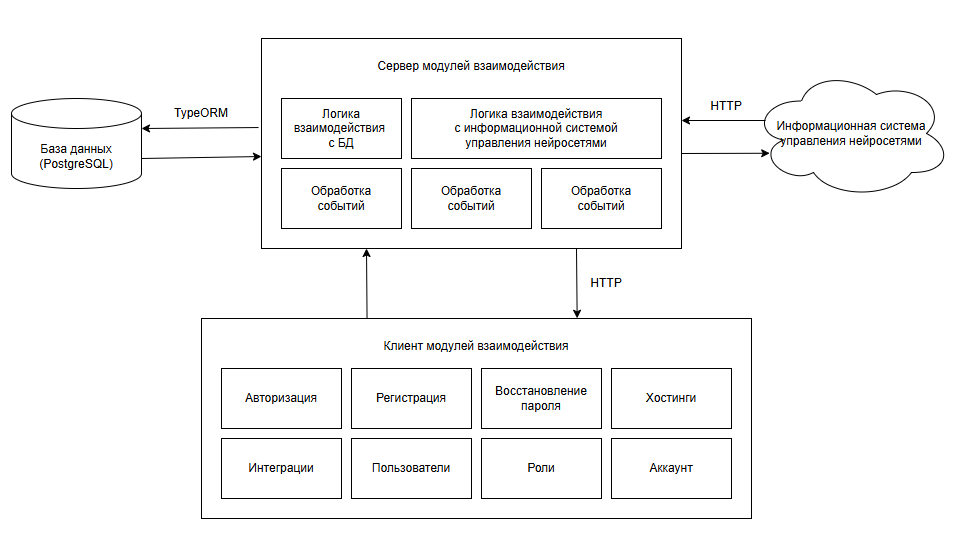


Рисунок А.10 – Диаграмма архитектуры взаимодействия

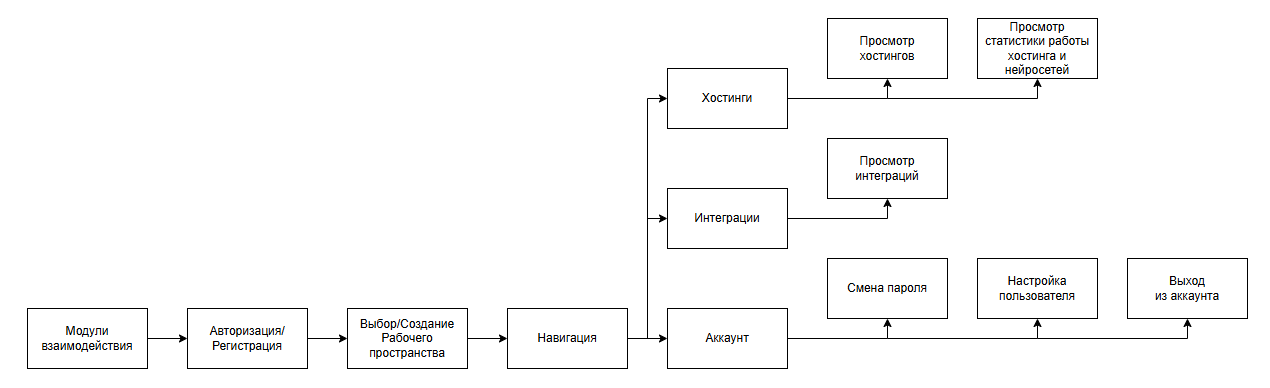


Рисунок А.11 - Диаграмма взаимодействия модулей для пользователя

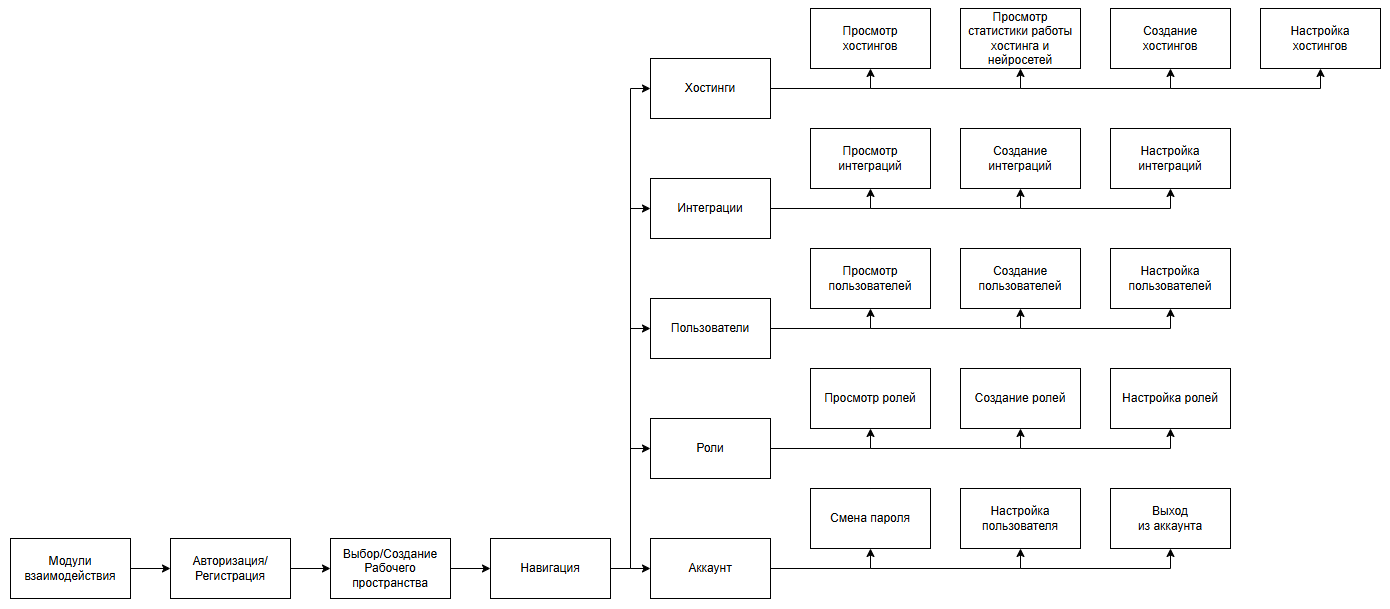


Рисунок А.12 - Диаграмма взаимодействия модулей для модератора

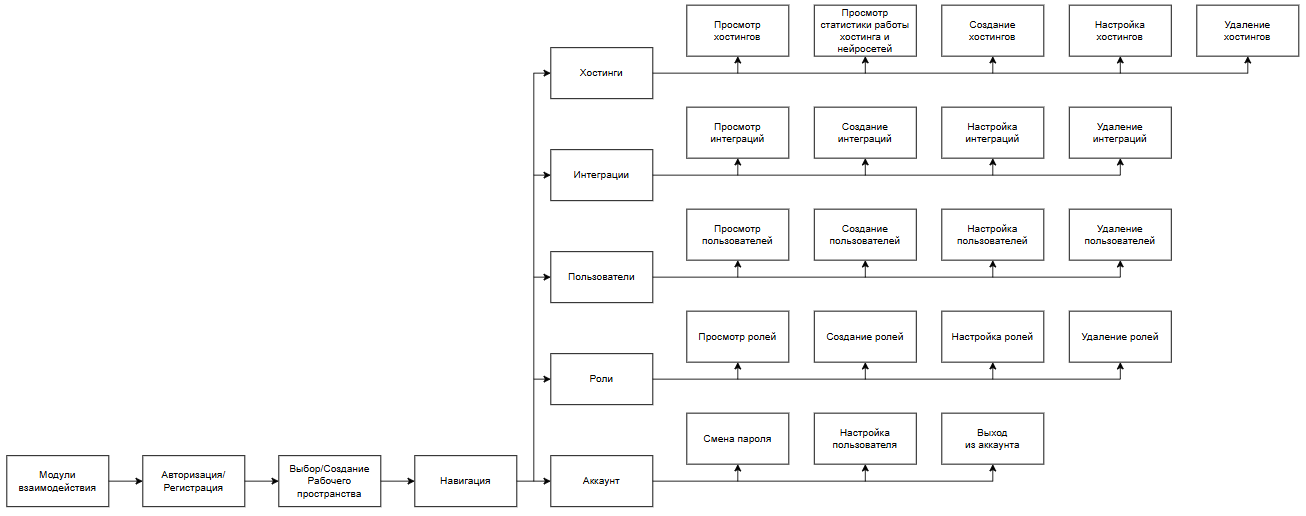


Рисунок А.13 - Диаграмма взаимодействия модулей для администратора

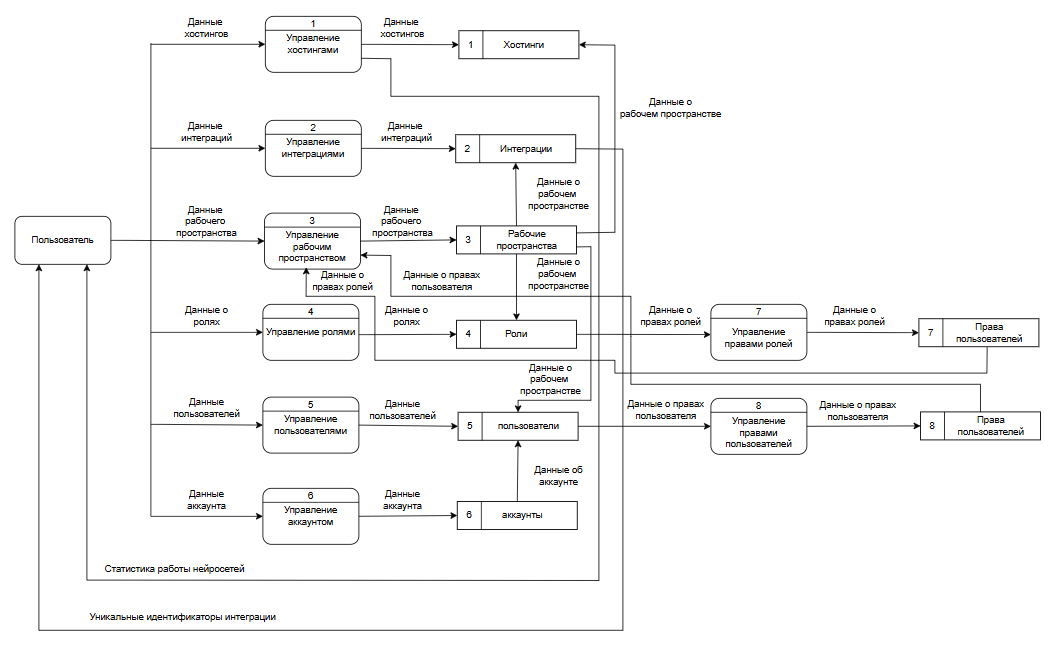


Рисунок А.4 – DFD модель

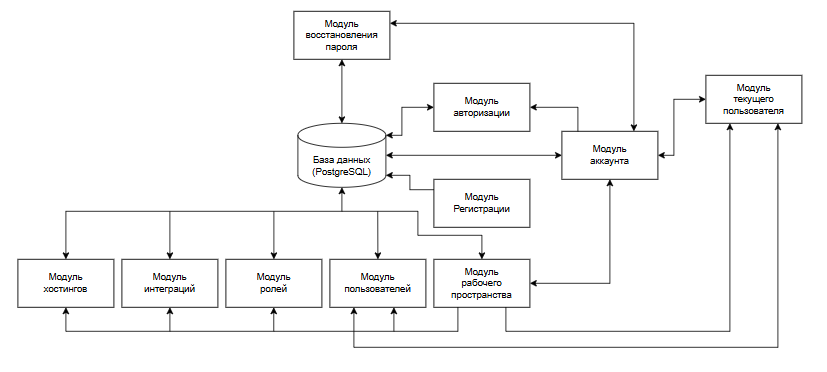


Рисунок А.15 - Диаграммы взаимодействия программ и программных модулей

# Приложение Б (Обязательное)

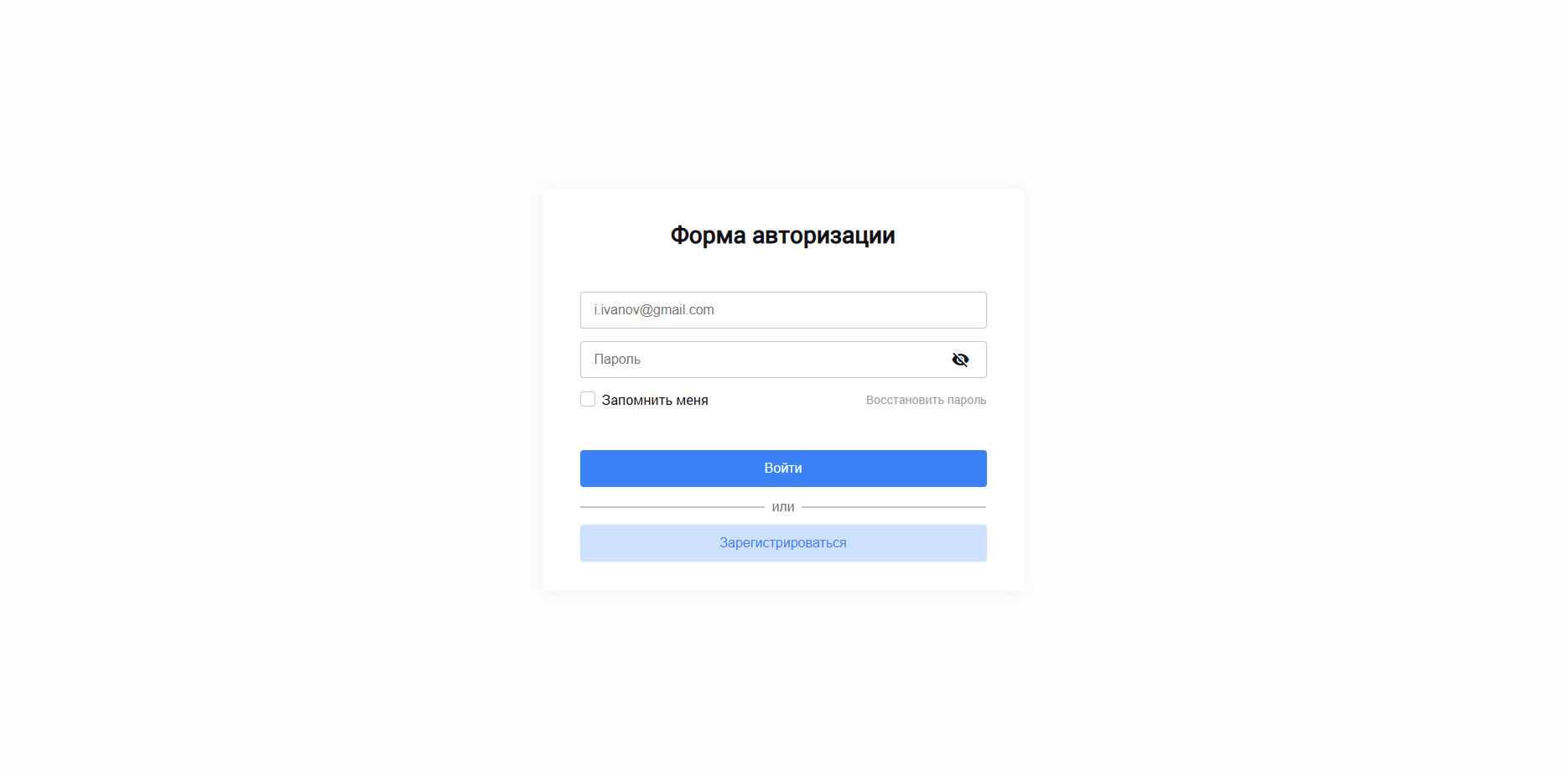


Рисунок Б.1 – Страница авторизации

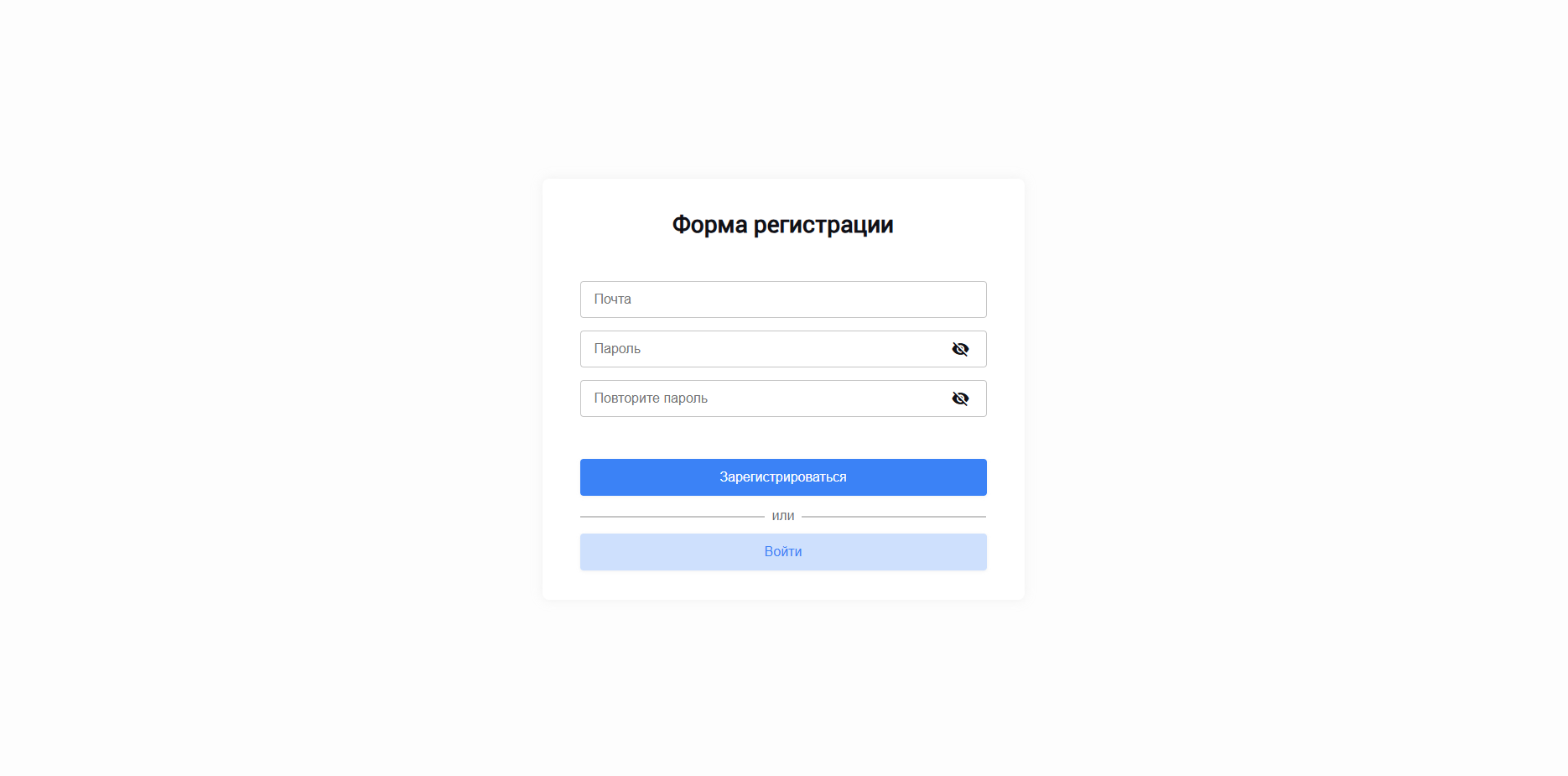


Рисунок Б.2 – Страница регистрации

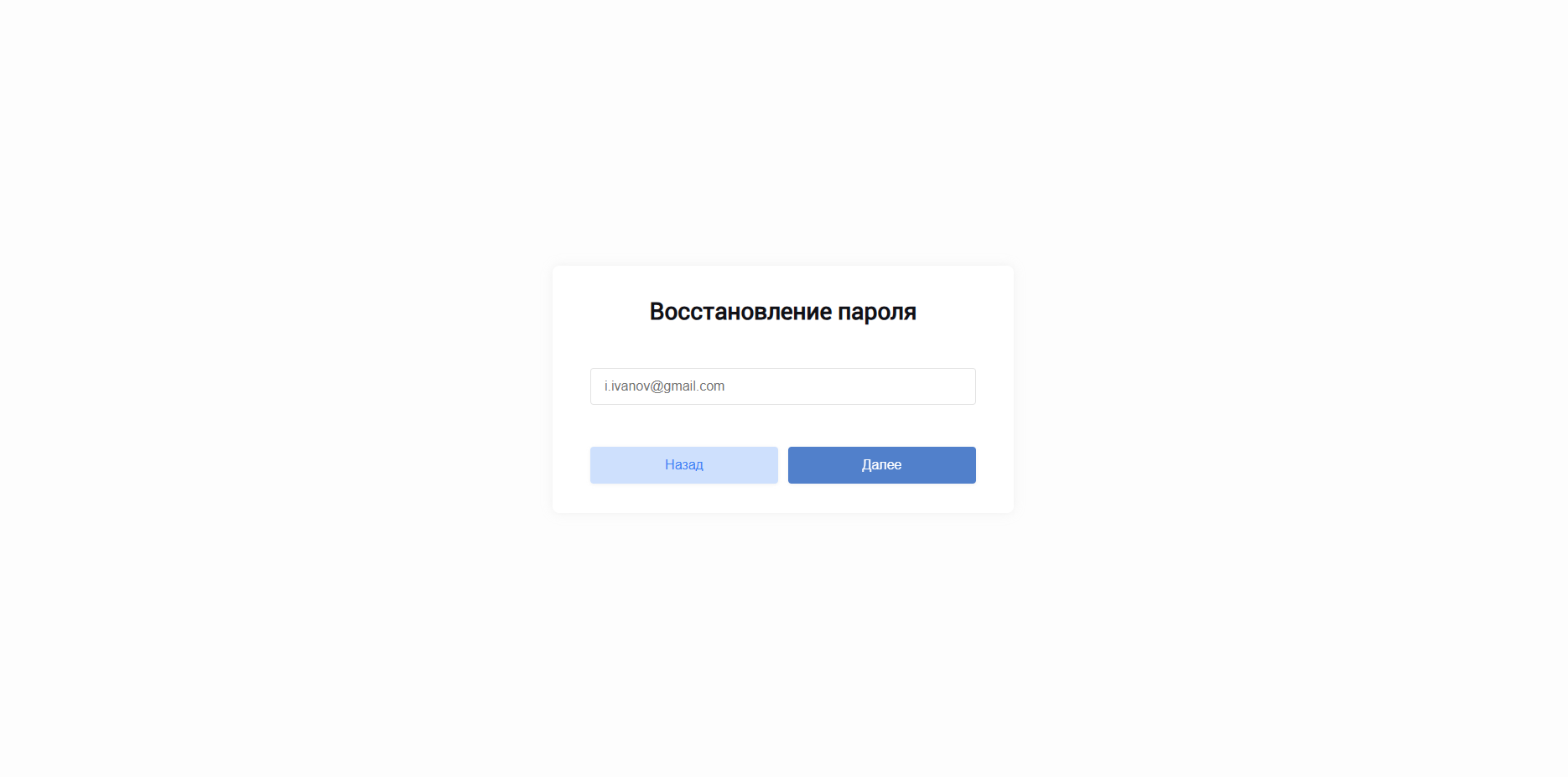


Рисунок Б.3 – Восстановление пароля

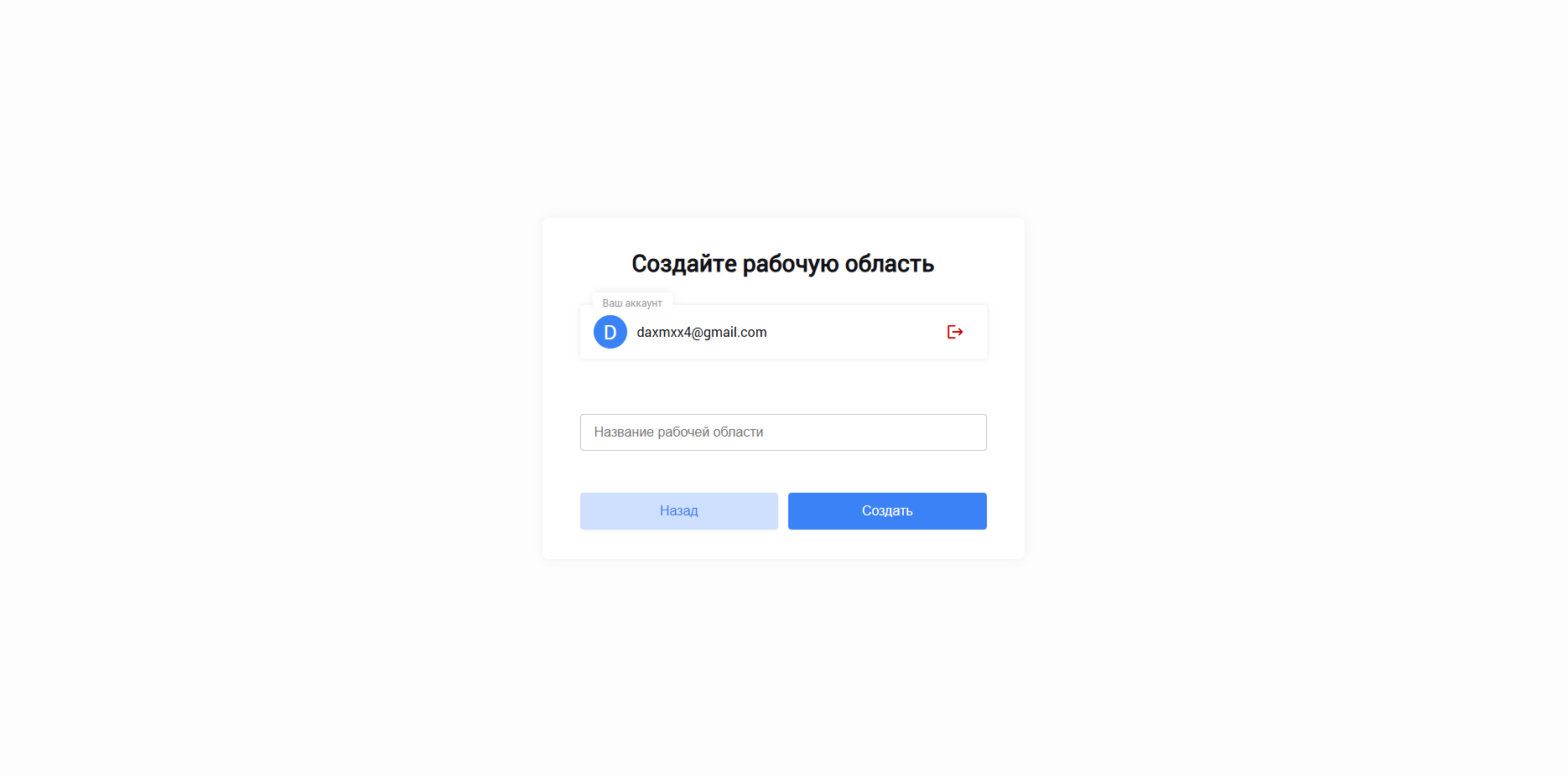


Рисунок Б.4 – Страница создания рабочей области

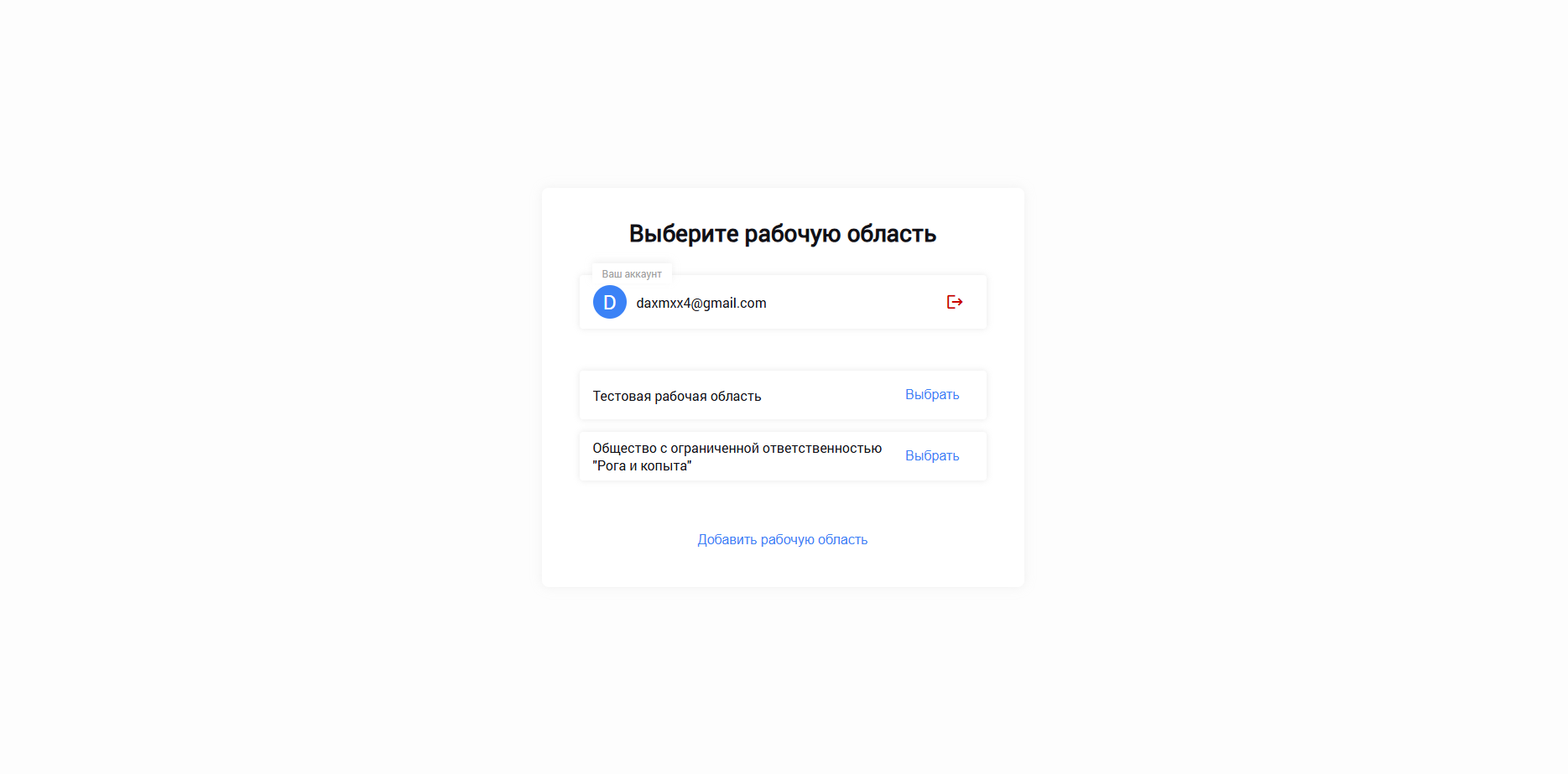


Рисунок Б.5 – Страница выбора рабочей области

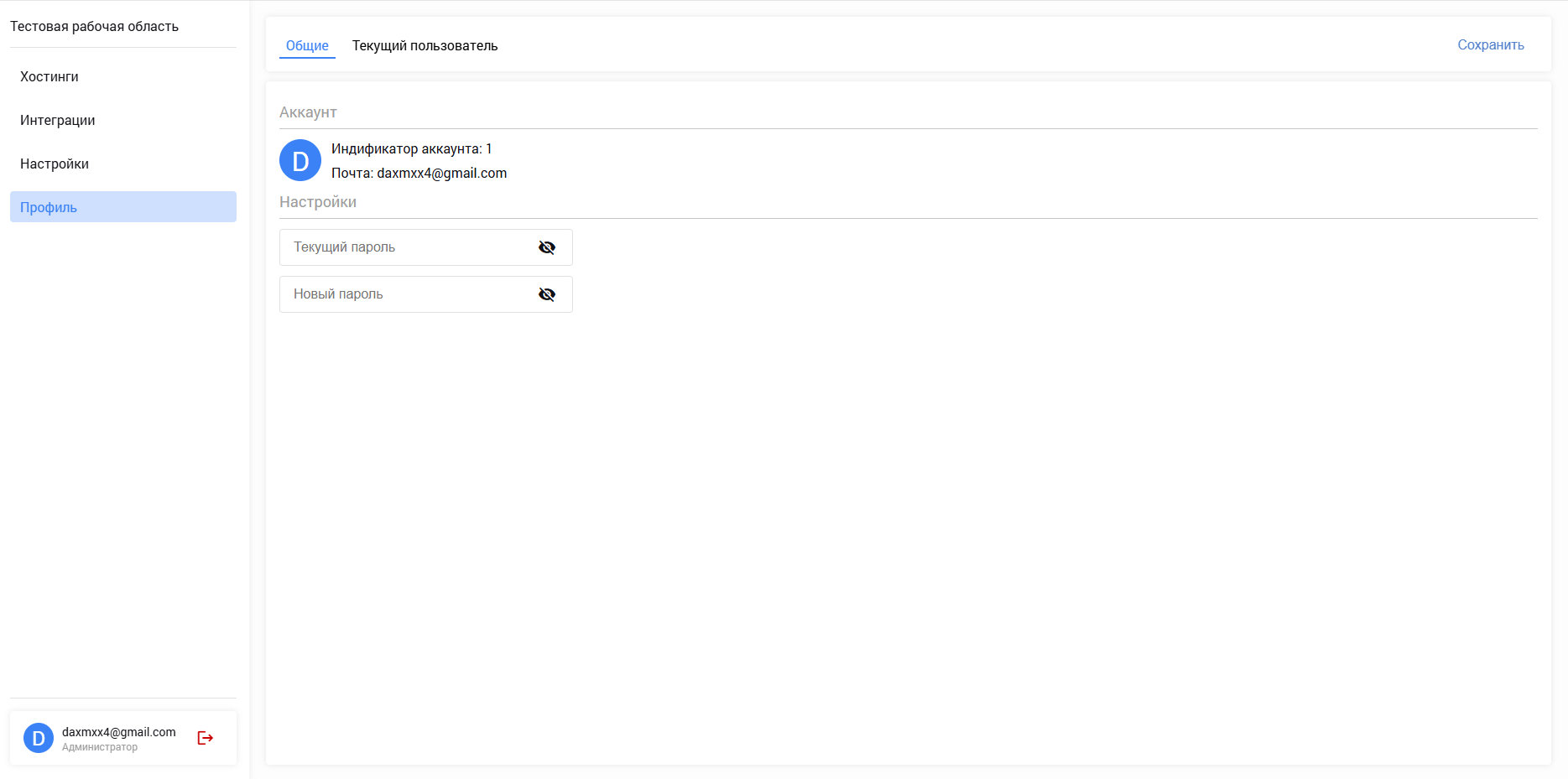


Рисунок Б.6 – Страница профиля, общие настройки

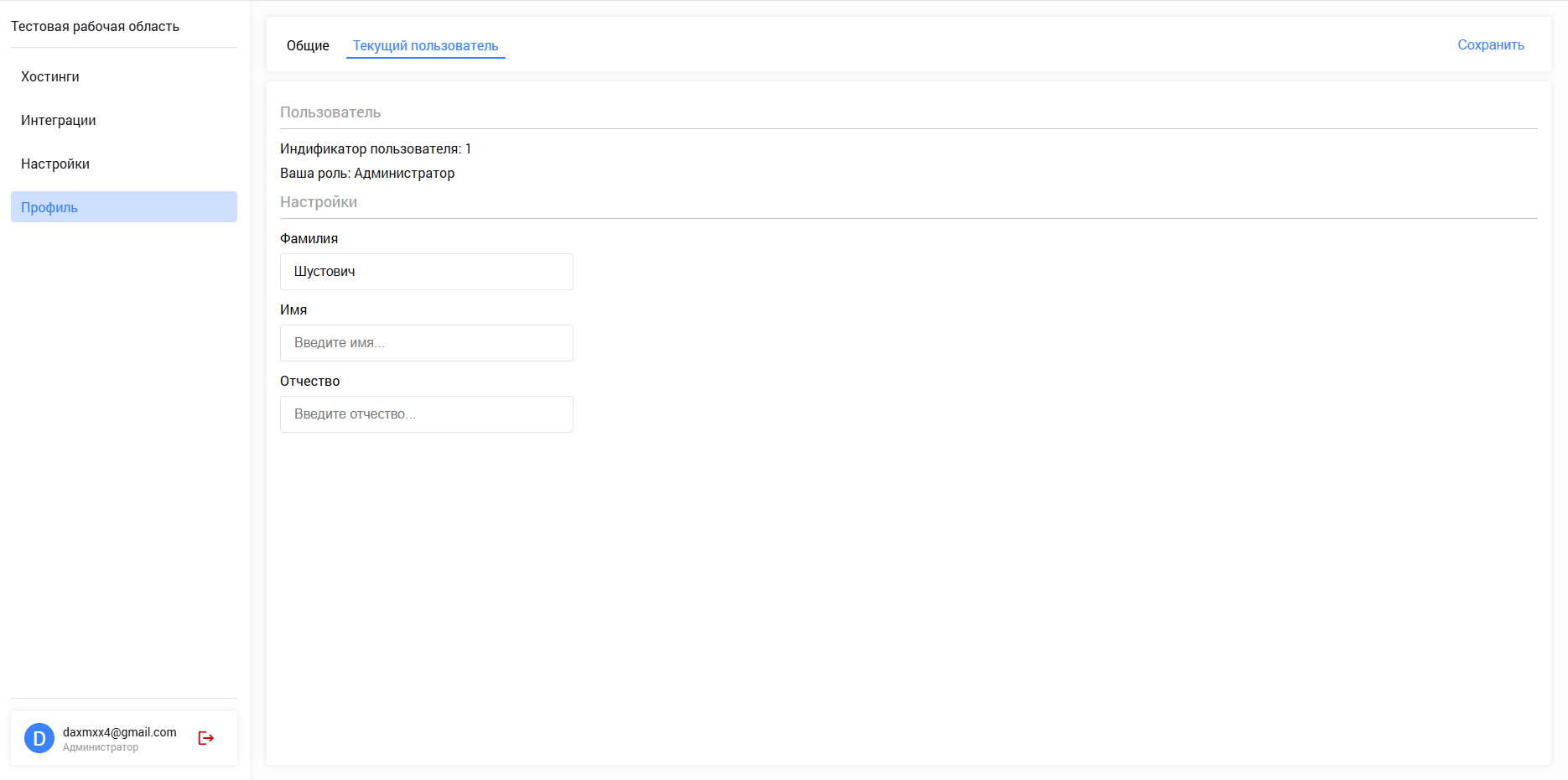
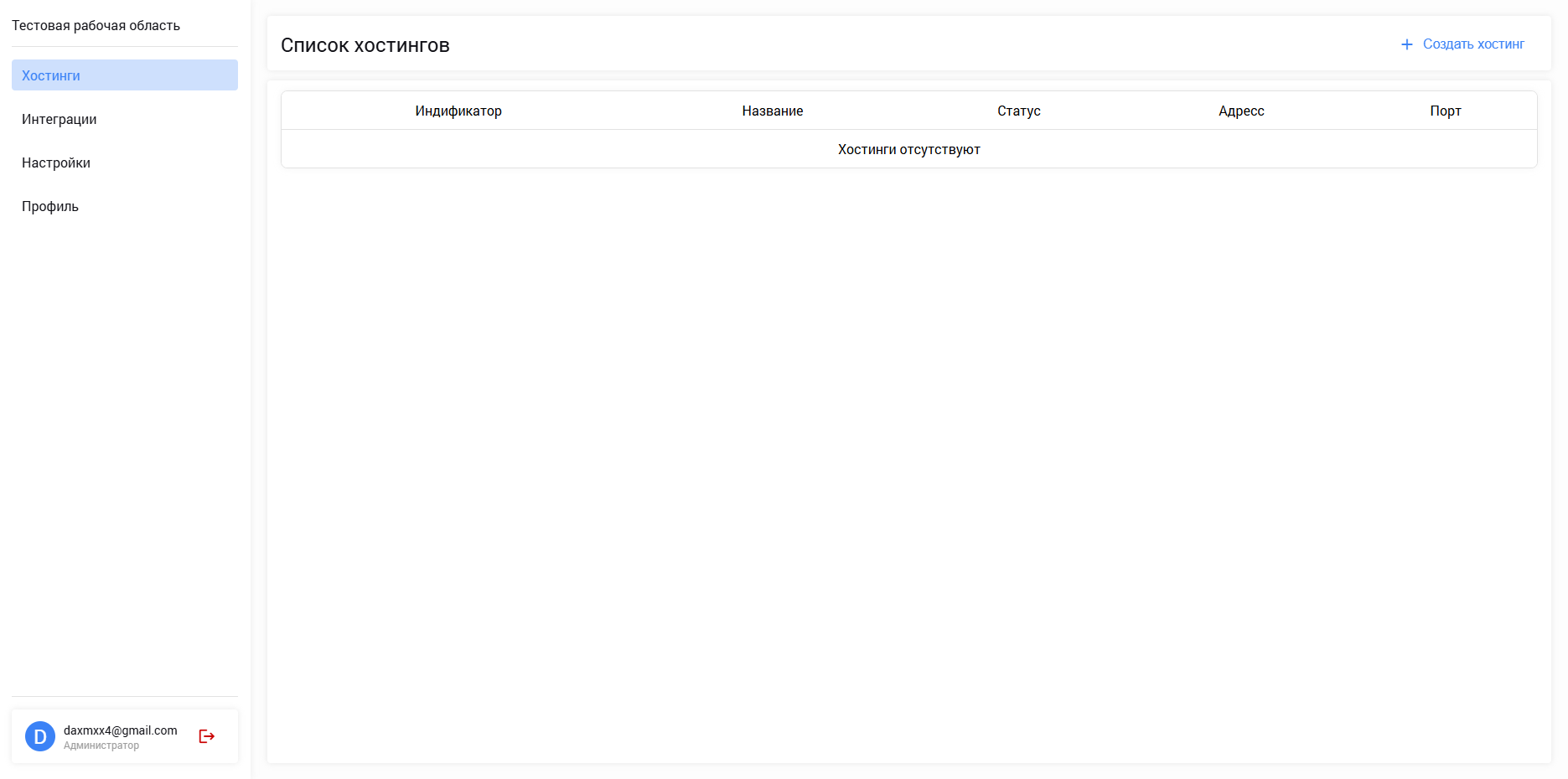
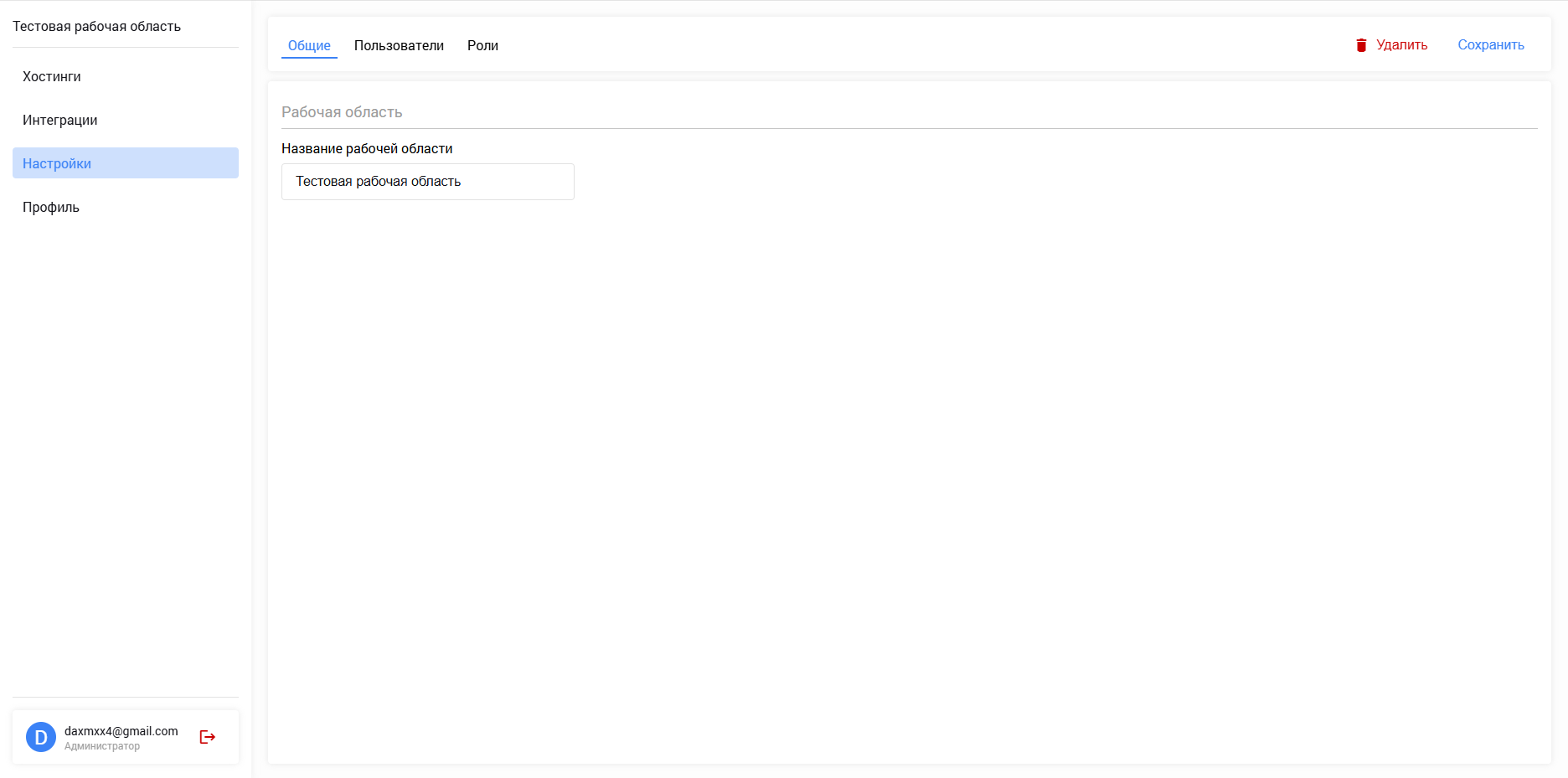
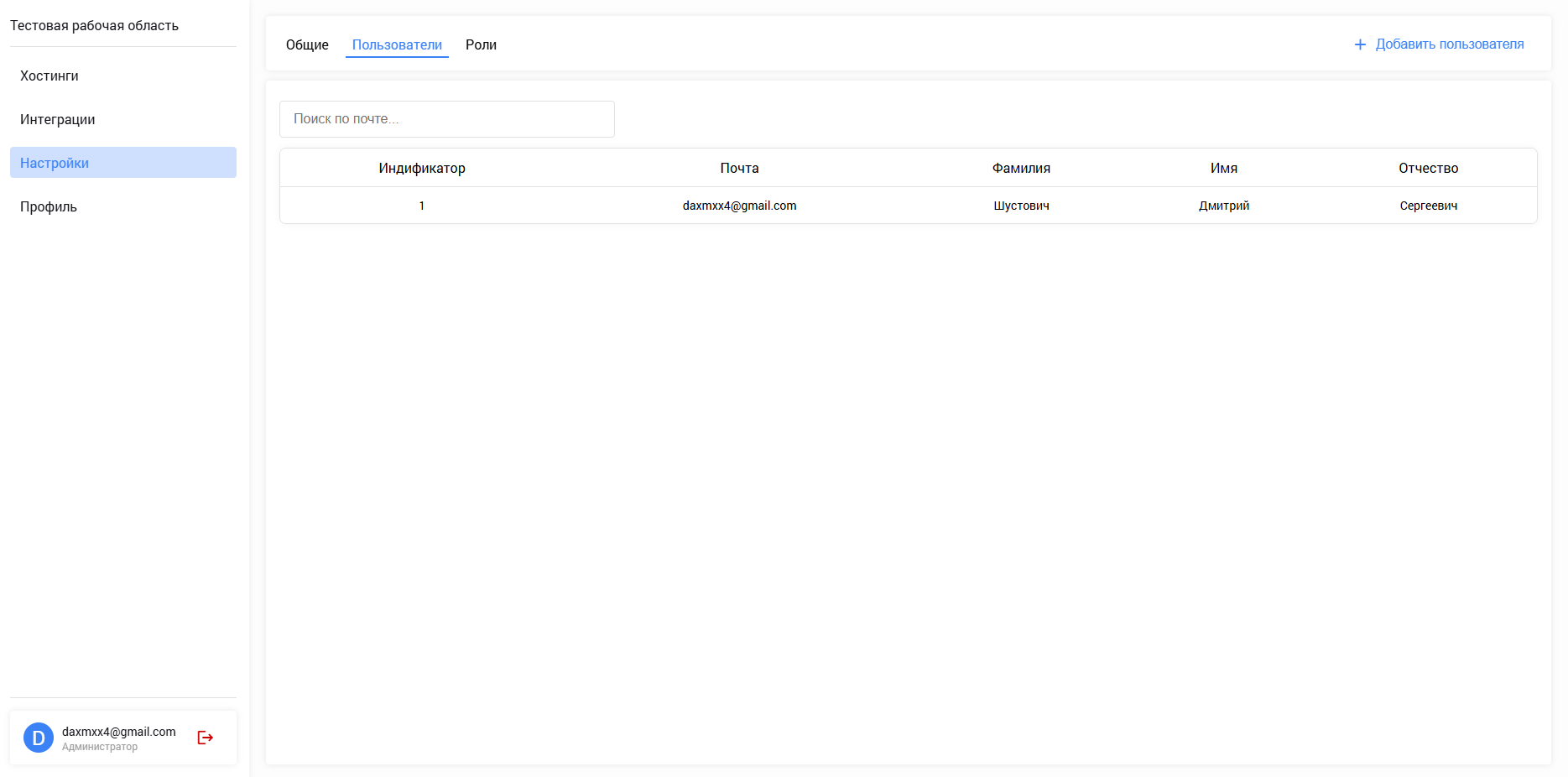


Рисунок Б.7 – Страница профиля, настройки текущего пользователя

  
Рисунок Б.8 – Страница списка хостингов

  
Рисунок Б.9 – Страница настроек рабочего пространства

  
Рисунок Б.10 – Страница настроек рабочего пространства, вкладка пользователей

## Приложение В (обязательное)



Рисунок В.1 – Тест-кейс для проверки функционала авторизации

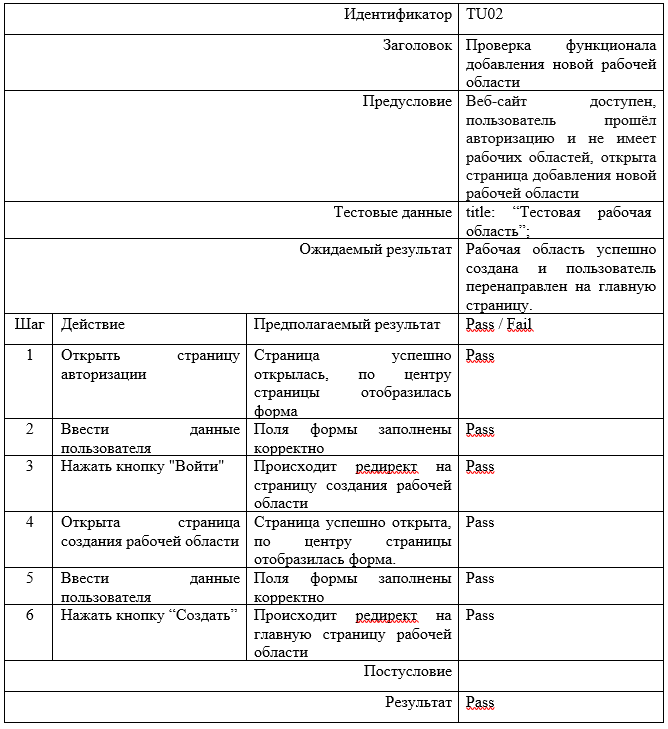


Рисунок В.2 – Тест-кейс для проверки функционала добавления рабочей области

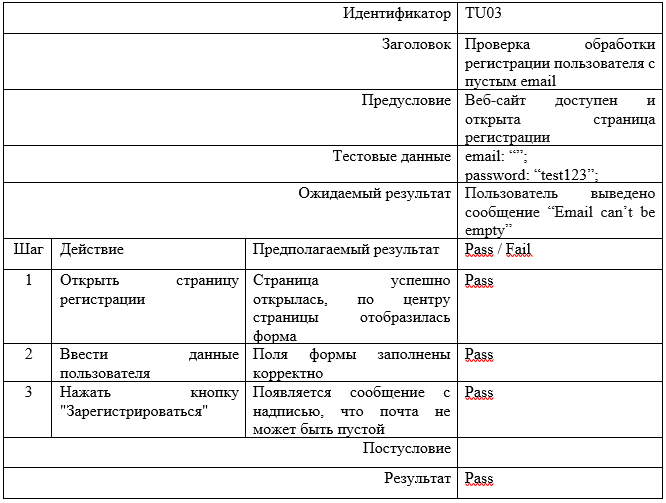


Рисунок В.3 – Тест-кейс для проверки обработки регистрации пользователя с пустым email

# Приложение Г (обязательное)

Г.1 Листинг блок-схемы алгоритма создания рабочего пространства

@Transactional()  
async create(dto: CreateWorkplaceDto): Promise<number> {  
 if (!dto.owner) {  
 throw new HttpException("Owner is null or undefined", HttpStatus.*BAD\_REQUEST*)  
 }  
  
 const workplace = this.rep.create(dto)  
  
 const result = await this.rep.insert(workplace)  
  
 workplace.id = result.identifiers[0].id  
  
 *// Создаём роли* await this.rolesService.initRolesByWorkplaceId(workplace.id)  
  
 *// Получаем роль администратора* const role = await this.rolesService.get(workplace.id, ERole.*ADMINSTRATOR*)  
  
 const user: CreateUserDto = {  
 workplace: workplace,  
 role: role,  
 account: {  
 id: dto.owner.id,  
 }  
 }  
  
 *// Создаём администратора в системе* await this.usersService.create(user)  
  
 return workplace.id  
}

Г.2 Листинг блок-схемы алгоритма создание хостинга

async create(dto: CreateHostingDto) {  
 const entity = this.rep.create(dto);  
  
 const connectToHostDto: ConnectToHostDto = {  
 address: dto.host,  
 port: dto.port,  
 description: dto.description  
 }  
 const [hosting, error] = await NNHHostingsApi.connectToHost(connectToHostDto)  
 if (!hosting || error) {  
 throw new HttpException(error?.message || "Unknow error", error?.status || HttpStatus.*BAD\_REQUEST*)  
 }   
 entity.id = hosting.id  
 entity.hostType = hosting.hostType  
 entity.status = hosting.status  
 entity.workplace.id = dto.workplace.id  
  
 return this.rep.save(entity)  
}