**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

**РОСАВИАЦИЯ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)**

**ФАКУЛЬТЕТ** Прикладной математики и вычислительной техники

**КАФЕДРА** Вычислительных машин, комплексов, систем и сетей

**Направление подготовки** 09.03.01, «Информатика и вычислительная техника»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

**(БАКАЛАВРСКАЯ РАБОТА)**

**Тема** «Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы»

**Обучающийся:**  Тропин Михали Денисович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

**Руководитель:**  \_ к.т.н., доцент, Терентьев А. И..\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(уч.степень, уч.звание, Ф.И.О.) (подпись)

**Работа допущена к защите:**

**Зав. Кафедрой**  д.т.н., профессор, Феоктистова О.Г. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(уч.степень, уч.звание, Ф.И.О.) (подпись)

**МОСКВА 2024**

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

-=BlackLynx=-

**РОСАВИАЦИЯ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**

**«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)**

Утверждаю Зав. кафедрой

Феоктистова О.Г.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 20 \_\_ г.

ФАКУЛЬТЕТприкладной математики и вычислительной техники

КАФЕДРА Вычислительных машин, комплексов, систем и сетей

Направление подготовки09.03.01, Информатика и вычислительная техника

(код, наименование направления подготовки)

**З А Д А Н И Е Н А Б А К А Л А В Р С К У Ю Р А Б О Т У**

**Обучающийся** Тропин Михаил Денисович

(фамилия, имя, отчество)

1. Тема работы \_ «Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы»\_

2. Утверждена приказом по университету от «12» декабря 2023 г. № 1209/ст

3. Срок сдачи обучающимся работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Исходные данные к работе

5. Содержание пояснительной записки (перечень подлежащих разработке вопросов) введение, анализ предметной области, анализ существующих систем, выбор технологических средств для реализации системы, техническое задание, разработка структуры информационных потоков системы, разработка алгоритмов работы системы, разработка структурной схемы системы, заключение, список используемых источников.

6. Перечень графического материала (с точным указанием обязательных чертежей) Постановка цели и задач (1л.), Выбор методов и средств разработки (1л.), Структура информационных потоков (1л.), Функциональная схема работы конфигуратора (1л.), Структура информационных потоков конфигуратора (1л.), алгоритм назначения прав(1л.), алгоритм авторизации (1л.), Заключение (1л.).

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № п/п | Наименование этапов выполнения ВКР | Срок выполнения этапов ВКР | Примечание |
| 1 | Анализ предметной области и обоснование необходимости разработки | 1 неделя |  |
| 2 | Написание технического задания | 1 неделя |  |
| 3 | Выбор методов и средств разработки конфигуратора | 1 неделя |  |
| 4 | Разработка структуры информационных потоков конфигуратора | 2 неделя |  |
| 5 | Разработка функциональной схемы работы конфигуратора | 2 неделя |  |
| 6 | Разработка алгоритмов работы системы | 2 неделя |  |
| 7 | Разработка интерфейса конфигуратора | 3 неделя |  |
| 8 | Разработка руководства пользователя | 4 неделя |  |
| 9 | Оформление пояснительной записки выпускной квалификационной работы | 4 неделя |  |

Задание выдал:

**Руководитель:** к.т.н., доцент, Терентьев А. И. \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(уч.степень, уч.звание, Ф.И.О.) (подпись)

Задание принял:

**Обучающийся:** Тропин Михаил Денисович \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (Ф.И.О. обучающегося) (подпись)

**АННОТАЦИЯ**

Целью выпускной квалификационной работы является сбор конкретного предметного материала для выполнения выпускной квалификационной работы на тему «Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы». Состоит из введения, 3 разделов, заключения, списка литературы. В работе использовано 17 рисунков, 3 приложения, 4 источников литературы. Общее количество страниц выпускной квалификационной работы 63.

Ключевые слова: конфигуратор, права доступа, клиент-серверная система.

Проведен анализ средств создания прикладного программного обеспечения, включая различные профессионально ориентированные программы, а также существующих решений с выявлением достоинств и недостатков.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ЭВМ-201002.ВКР.1209.22.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Тропин М. Д. |  |  | Аннотация | Литера | | | Лист | Листов |
| Пров | | Терентьев А. И. |  |  |  |  |  | 4 | 1 |
|  | |  |  |  | МГТУ ГА ФПМ и ВТ  ЭВМ 4-1 090301 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Феоктистова О.Г. |  |  |

СОДЕРЖАНИЕ

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ЭВМ-201002.ВКР.1209.22.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Тропин М. Д. |  |  | Содержание | Литера | | | Лист | Листов |
| Пров | | Терентьев А. И. |  |  |  | у |  | 5 | 2 |
|  | |  |  |  | МГТУ ГА ФПМ и ВТ  ЭВМ 4-1 090301 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Феоктистова О.Г. |  |  |

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 6](#_Toc170146681)

[ВВЕДЕНИЕ 8](#_Toc170146682)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc170146683)

[1.1 Анализ существующих решений 13](#_Toc170146684)

[1.2 Структурная схема конфигуратора 15](#_Toc170146685)

[1.3 Средства и технологии программирования 17](#_Toc170146686)

[1.3.1 Выбор методов разработки, хранения и управления данными 17](#_Toc170146687)

[1.3.2 Выбор среды разработки 17](#_Toc170146688)

[1.3.3 Выбор языка программирования 19](#_Toc170146689)

[2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 21](#_Toc170146690)

[2.1 Общие сведения 21](#_Toc170146691)

[2.2 Цели и назначение создания конфигуратора 21](#_Toc170146692)

[2.3 Требования к конфигуратору 21](#_Toc170146693)

[3. РАЗРАБОТКА КОНФИГУРАТОРА УПРАВЛЕНИЯ ПРАВАМИ ПОЛЬЗОВАТЕЛЕЙ 24](#_Toc170146694)

[3.1 Разработка структуры информационных потоков конфигуратора управления правами пользователей 24](#_Toc170146695)

[3.2 Разработка функциональной схемы работы конфигуратора 26](#_Toc170146696)

[3.3 Разработка алгоритмов работы конфигуратора 28](#_Toc170146697)

[3.4 Разработка структуры веб-сайта конфигуратора 32](#_Toc170146698)

[3.5 Разработка интерфейса веб-сайта конфигуратора 33](#_Toc170146699)

[ЗАКЛЮЧЕНИЕ 36](#_Toc170146700)

[СПИСОК ЛИТЕРАТУРЫ 37](#_Toc170146701)

[ПРИЛОЖЕНИЕ А 38](#_Toc138017694)

[ПРИЛОЖЕНИЕ Б 39](#_Toc138017695)

[ПРИЛОЖЕНИЕ В 55](#_Toc138017696)

*Изм. Изм.*

*Лист Лист*

*№ докум.*

*Подпись*

*Дата*

Лист

*6*

ЭВМ-201002.ВКР.1209.22.ПЗ

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

МО — Московская область;

РНИС — Региональная навигационно-информационная система;

СТЕНД — набор виртуальных машин привязанных к определенному региону.

# ВВЕДЕНИЕ

Современный мир информационных технологий и цифровых коммуникаций требует новых подходов к управлению бизнесом и обслуживанию клиентов. С каждым годом индустрия и рынок IT развивается с огромной скоростью. Штабы компаний предоставляющих цифровые информационные услуги продолжают расширятся. Количество данных и, обрабатывающих их, сотрудников растет.

Существует множество способов распределения прав просмотра и редактирования сотрудниками различных баз данных. Основным способом является назначение прав владельцем базы данных через запросы в системы управления баз данных, что не всегда является быстрыми и удобным решением. В то время как создание такого инструмента во многом позволит качественно упростить и оптимизировать процесс работы пользователям системы РНИС.

Корпоративная система РНИС (Единая региональная навигационно-информационная система) работает с множеством баз данных и большим количествам сотрудников, которым нужно распределить доступ к данным для решения определенных задач. При этом невозможно назначить права сразу на все базы данных, в целях защиты данных от непреднамеренных изменений.

На основании вышеизложенного, можно сделать вывод, что создание данного конфигуратора управления правами пользователей позволит улучшить скорость и эффективность работы компании.

Таким образом, разрабатываемый «Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы» бесспорно актуален.

## **1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Региональная навигационно-информационная система (РНИС) — комплексное решение, которое используется для управления транспортной инфраструктурой, включая дороги, мосты, тоннели, аэропорты, железнодорожные станции и другие объекты. На рисунке 1.1 представлен интерфейс портала РНИС МО.

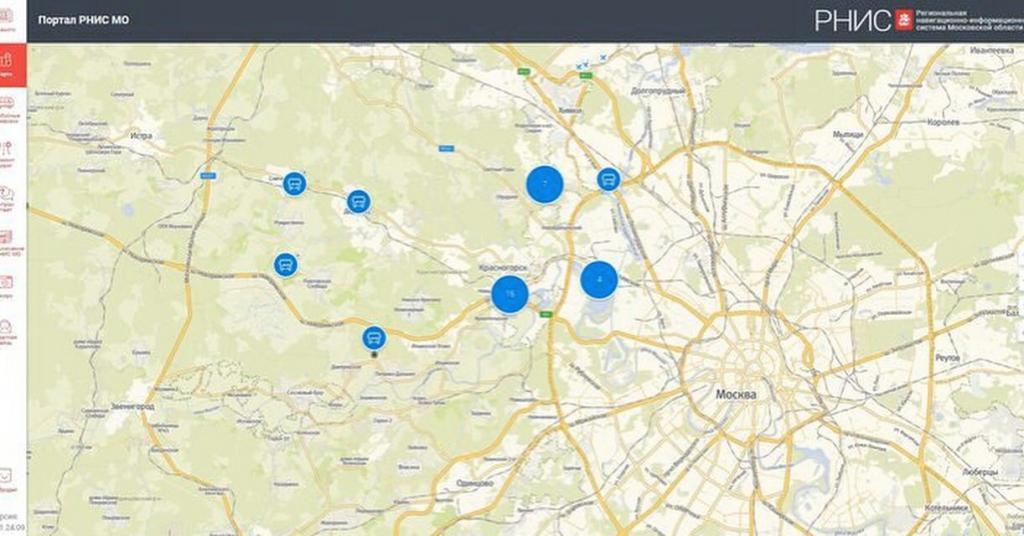


Рисунок 1.1 — Интерфейс портала РНИС МО

Инфраструктура РНИСа включает в себя несколько компонентов, включая серверы, базы данных, сетевое оборудование, программное обеспечение и другое.

Инфраструктурная поддержка РНИСа включает в себя ряд задач, связанных с поддержкой и обслуживанием инфраструктуры, которые могут быть разделены на следующие категории:

Управление серверами и базами данных. Эта категория включает в себя управление серверами и базами данных, мониторинг и управление ресурсами, создание резервных копий и восстановление данных, обновление программного обеспечения и операционных систем, управление уязвимостями и другие задачи, связанные с управлением серверами и базами данных.

Управление сетевым оборудованием. Эта категория включает в себя управление сетевым оборудованием, мониторинг сетевых устройств и их производительности, управление настройками сетевых устройств, обнаружение и устранение сбоев в сети, управление уязвимостями и другие задачи, связанные с управлением сетевым оборудованием.

Управление программным обеспечением. Эта категория включает в себя управление программным обеспечением РНИС, установка и настройка программного обеспечения на серверах и рабочих станциях, мониторинг и управление производительностью программного обеспечения, обновление программного обеспечения и другие задачи, связанные с управлением программным обеспечением.

Управление безопасностью. Эта категория включает в себя управление безопасностью инфраструктуры РНИС, мониторинг угроз безопасности, создание и обновление политик безопасности, управление уязвимостями, обеспечение соответствия требованиям безопасности и другие задачи, связанные с управлением безопасностью.

Управление ресурсами. Эта категория включает в себя управление ресурсами инфраструктуры РНИС, включая управление оборудованием, программным обеспечением, сетевыми ресурсами, управление электропитанием, управление охлаждением и другие задачи, связанные с управлением ресурсами.

Для обеспечения эффективной инфраструктурной поддержки РНИСа необходимо наличие высококвалифицированных специалистов, готовых решать множество задач, связанных с управлением инфраструктурой. Также важно иметь комплексы автоматизированного многофункционального программного обеспечения для мониторинга, управления ресурсами, управления уязвимостями и другие инструменты, которые позволяют эффективно управлять инфраструктурой и обеспечивать ее безопасность и производительность. Одним из таких является инструмент - конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы.

В работе рассматривается разработка конфигуратора управления правами в базах данных для пользователей системы РНИС (Единая региональная навигационно-информационная система).

Основной задачей конфигуратора является предоставление пользователям, ответственным за распределение прав, удобного графического инструмента.

На данный период штат компании превышает сотню человек, и делится на 5 отделов:

* Отдел разработки php
* Отдел разработки go
* Отдел разработки интерфейса пользователей
* Отдел техподдержки
* Отдел аналитики

Права сотрудники зависят как от отделов, так и от их должности в отделе.

**Потенциальные пользователи**

Единственными пользователями инструмента могут быть только сотрудниками третьего уровня техподдержки. Остальные сотрудники для получения прав на интересующие их данные должны обратится к сотруднику третьего уровня техподдержки.

**Необходимость создания инструмента назначения прав**

Инструмент разрабатывается с целью оптимизации времени, уходящего у сотрудника техподдержки на назначение прав сотрудникам.

**Преимущества использования инструмента назначения прав**

Использование инструмента назначения прав имеет следующие преимущества:

* Уменьшение затрат времени сотрудника на назначение прав
* Повышение уровня удобства сотрудников.

Таким образом, анализ предметной области " Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы" показывает, что такая система является необходимой для обеспечения пользователей нужным и удобным инструментом. Он позволит компании сохранять высокие стандарты скорости и качества, что является ключевым фактором для работы системы РНИС.

# 

# 1.1 Анализ существующих решений

**СУБД**

PostgreSQL, или Postgres — объектно-реляционная система управления базами данных (СУБД), которая использует и расширяет язык SQL. Postgres была разработана в Университете Калифорнии в Беркли в 1986 году и является одной из самых мощных и расширяемых СУБД в мире.

На рисунке 1.2 представлен логотип СУБД PostgreSQL. 

Рисунок 1.2 — Логотип СУБД PostgreSQL

Postgres имеет множество функций, которые делают его одним из наиболее гибких и мощных решений для управления базами данных.

Поддержка многопоточности: Postgres поддерживает многопоточность, что позволяет использовать несколько ядер процессора для обработки запросов.

Расширяемость: Postgres имеет множество расширений и плагинов, которые могут быть использованы для расширения функциональности базы данных.

Поддержка объектов: Postgres поддерживает объекты, такие как таблицы, представления, индексы и многое другое, что обеспечивает гибкость в управлении данными.

Транзакции: Postgres поддерживает транзакции, что позволяет выполнить несколько запросов как единое целое и откатить их в случае неудачи.

Репликация: Postgres поддерживает репликацию, что позволяет создавать копии базы данных для обеспечения отказоустойчивости и увеличения производительности.

Безопасность: Postgres имеет множество функций для обеспечения безопасности данных, такие как шифрование, аутентификация и авторизация.

Postgres является одной из самых мощных и расширяемых СУБД в мире. Он предоставляет широкий набор функций для управления данными, отказоустойчивости и безопасности. Он может быть использован в любом проекте, который требует управления данными.

Недостаток: Назначение пользователям прав в Postgres осуществляется с помощью SQL запросов, что является не самым удобным способом, так как на написание запросов нужно тратить время. Также большая часть времени тратится на переключение между БД при необходимости выполнять запросы в нескольких БД. Еще к недостаткам можно отнести то, что не для каждого будет удобен интерфейс.

# 1.2 Структурная схема конфигуратора

Первым этапом разработки конфигуратора управления правами пользователей является создание её структурной схемы, отражающей все основные компоненты, которые необходимы для реализации.

На рисунке 1.3 представлена структурная схема конфигуратора управления правами пользователей.

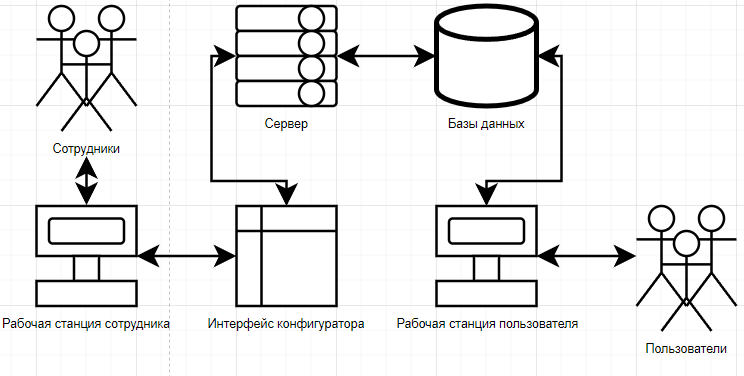


Рисунок 1.3 — Структурная схема конфигуратора управления правами пользователей

Схема описывает взаимодействие пользователей и сотрудников с сервером через интерфейс, а также взаимодействие сервера с базой данных.

Схема состоит из:

- Сотрудников, которые назначают права;

- Рабочей станции сотрудника, на которой он взаимодействует с интерфейсом конфигуратора;

- Пользователей, которым назначают права;

- Рабочей станции пользователя, с которой он взаимодействует с СУБД;

- Интерфейс. Его функция обеспечивать связь пользователей с СУБД;

- Сервер. Главная функция – обработка запросов от интерфейса и обращения к базе данных;

- База данных. Хранит данные о заявках и пользователях.

Конфигуратор состоит из двух основных частей:

* Пользовательская часть. Пользователи взаимодействуют с пользовательским интерфейсом, который позволяет авторизоваться и распределять права на web-страницах;
* Серверная часть. Сервер посылает запросы к базе данных на получение, изменение и удаление данных. Также там хранятся web-страницы и скрипты, необходимые для функционирования конфигуратора. Сервер базы данных хранит все необходимые данные.

# 1.3 Средства и технологии программирования

## **1.3.1 Выбор методов разработки, хранения и управления данными**

Все необходимые действия с данными будут производиться с использованием БД Postgres при помощи разрабатываемого веб-интерфейса.

Postgres имеет множество программных интерфейсов (API), благодаря которым к базе данных Postgres могут подключаться приложения, созданные с помощью C/C++, Eiffel, Java, Perl, PHP, Python, ODBC, NET и Visual Studio.

## **1.3.2 Выбор среды разработки**

Для создания конфигуратора управления правами будет создан web-сайт. Сайт будет создан вручную, без использования различных конструкторов. При создании конфигуратора буду использованы:

Язык HTML — (от англ. HyperText Markup Language — «язык гипертекстовой разметки») — это основной строительный блок Интернета. Он определяет значение и структуру веб-контента. Другие технологии, помимо HTML, обычно используются для описания внешнего вида/представления веб-страницы (CSS) или функциональности/поведения (JavaScript) [1].

На рисунке 1.4 представлен логотип HTML.



Рисунок 1.4 — Логотип HTML

Большинство веб-страниц написано в документах HTML и интерпретируется браузерами, клиентами. Интерпретированный текст затем отображается на экране компьютера или мобильного устройства. HTML позволяет легко создавать относительно простые, но красиво оформленные веб-сайты.

Разметка web-сайта конфигуратора разработана с использованием HTML — стандартизированным языком гипертекстовой разметки документов для просмотра веб-страниц в браузере, а дизайн с использованием CSS — формальный язык декодирования и описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки.

Visual Studio Code — бесплатный и открытый исходный редактор кода, который был создан компанией Microsoft. Он представляет собой легковесный, кроссплатформенный инструмент для разработки, который поддерживает множество языков программирования и интегрируется с различными инструментами разработки, такими как Git, IntelliSense, отладчик и другие.

На рисунке 1.5 представлен логотип среды разработки Visual Studio Code

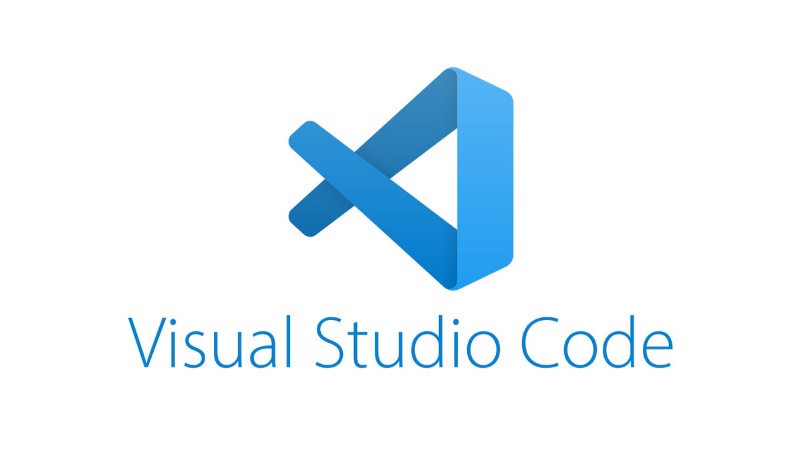


Рисунок 1.5 — Логотип среды разработки Visual Studio Code

Visual Studio Code используется для разработки программного обеспечения, веб-приложений и сайтов, а также для написания скриптов и много другого. Он имеет множество функций, которые помогают разработчикам ускорить процесс разработки, такие как авто дополнение кода.

GitLab — это инструмент управления исходным кодом, который обеспечивает полный цикл разработки программного обеспечения. Это открытый исходный код, который может быть развернут на собственных серверах, или использован как облачный сервис. GitLab поддерживает Git, систему контроля версий, и предоставляет широкий спектр функций для управления исходным кодом, отслеживания ошибок, автоматизации сборки и развертывания, до управления запросами на слияние и тестирования кода [2].

На рисунке 1.6 представлен логотип инструмента управления исходным кодом GitLab.



Рисунок 1.6 — Логотип инструмента управления исходным кодом GitLab

Некоторые из ключевых функций GitLab включают в себя:

1. Хостинг репозиториев Git: GitLab предоставляет возможность создания и хостинга репозиториев Git, что позволяет управлять исходным кодом проекта.
2. Управление запросами на слияние: GitLab позволяет управлять запросами на слияние, что упрощает процесс совместной работы над проектом.
3. Отслеживание ошибок: GitLab предоставляет инструменты для отслеживания ошибок и управления задачами, что помогает организовать процесс разработки.
4. Интеграция с CI/CD: GitLab интегрируется с системами непрерывной интеграции и непрерывной доставки (CI/CD), что позволяет автоматизировать процесс сборки, тестирования и развертывания приложения.

## **1.3.3 Выбор языка программирования**

Для разработки конфигуратора был использован язык программирования Java Squirt.

JavaScript — мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией стандарта ECMAScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений [3].

На рисунке 1.7 - представлен логотип JavaScript.



Рисунок 1.7 — Логотип JavaScript

Node JS — это не отдельный язык программирования, а среда для выполнения JavaScript. Или, другими словами, платформа для работы с JavaScript на стороне сервера. Раньше JavaScript мог выполнять только браузер. С помощью Node.js это можно делать без него [4].

Чаще всего Node используют для написания серверной части сайта, однако сейчас на нём пишут и десктопные программы.

На рисунке 1.8 - представлен логотип JavaScript.

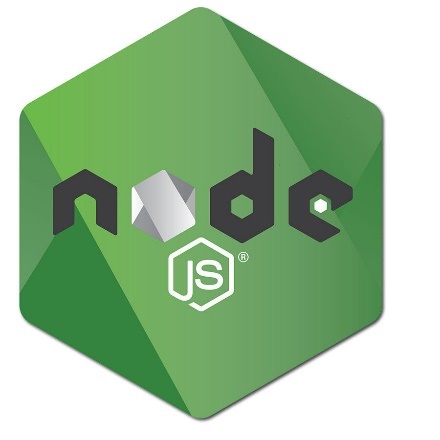


Рисунок 1.8 — Логотип Node JS

# 2. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## **2.1 Общие сведения**

**Введение**

В рамках данной выпускной квалификационной работы разрабатывается конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы.

**Основание для разработки конфигуратора**

Тема выпускной квалификационной работы «Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы» утверждена приказом по университету (Московский Государственный Технический Университет Гражданской Авиации) от «12» декабря 2023г. № 1209/ст.

## **2.2 Цели и назначение создания конфигуратора**

**Назначение разработки**

Разрабатываемый конфигуратор позволит повысить скорость и удобство назначения прав пользователей к базам данных.

**Цель разработки конфигуратора**

Целью разработки конфигуратора является:

* Минимизировать затрачиваемые временные ресурсы;
* Упрощение процесса назначения прав пользователей;

## **2.3 Требования к конфигуратору**

**Функциональные требования**

* Разрабатываемый конфигуратор должен быть Web–сайтом;
* Конфигуратор должен быть масштабируемый и готовый к добавлению новых функций в будущем;
* Конфигуратор должен представлять собой программу с адаптивным интуитивно понятным графическим интерфейсом пользователя;
* Обмен данными между компонентами должен быть организован с помощью протокола HTTPS;
* Предусмотрена возможность расширяемости программного продукта;
* Разрабатываемый конфигуратор должен предоставлять сотруднику возможность назначить права пользователю на просмотр, изменения и удаление данных в базах данных, и возможность создания и удаления пользователей;
* Должна быть реализована авторизация пользователя, для доступа к инструменту только строго определенных пользователей.

**Требования к программному изделию**

* Интерфейс должен быть интуитивно понятным, не требовать много времени на изучение;
* Пользователь должен получать наглядное представление о том, какой функционал несет страница;
* Информация на навигационных элементах должна иметь однозначное толкование.

**Требования к информационному обеспечению**

* Для реализации статических страниц и шаблонов должны использоваться языки HTML, CSS, JS;
* Распределение прав пользователей будет проходить в СУБД Postgres;
* Программное обеспечение должно обеспечивать надежное хранение данных и защиту от несанкционированного доступа;
* Программное обеспечение должно быть масштабируемым и готовым к добавлению новых функций в будущем;
* Программное обеспечение должно обеспечивать высокую производительность и быстродействие;
* Программное обеспечение должно быть отказоустойчивым и способным восстанавливаться в случае сбоев в работе;

**Требования к аппаратным средствам**

* Устойчивое функционирование аппаратных средств;
* Бесперебойное питание аппаратных средств.

**Стадии и этапы разработки**

* Анализ предметной области и обоснование необходимости разработки конфигуратора;
* Проектирование структурной схемы конфигуратора;
* Выбор методов разработки конфигуратора;
* Разработка структуры информационных потоков и алгоритмов работы конфигуратора;
* Разработка пользовательского интерфейса конфигуратора;
* Отладка и тестирование конфигуратора;
* Оформление программной документации.

**Требования к количеству и квалификации персонала**

Уверенный пользователь ПК.

**Требования к составу и параметрам технических средств**

Минимальные требования для работы программного средства:

* Операционная система Windows 7/Mac OS/ Linux;
* Наличие выхода в сеть Интернет;
* Клавиатура и мышь либо сенсорный экран.

**Требования к маркировке и упаковке**

Не предъявляются.

**Требования к транспортированию и хранению**

Не предъявляются.

# 3. РАЗРАБОТКА КОНФИГУРАТОРА УПРАВЛЕНИЯ ПРАВАМИ ПОЛЬЗОВАТЕЛЕЙ

# 3.1 **Разработка структуры информационных потоков конфигуратора управления правами пользователей**

Структура информационных потоков конфигуратора состоит из двух частей. Первая часть является пользовательской, именно с ней работает конечный пользователь. Вторая часть – серверная, выполняющая все операции с поступающей информацией.

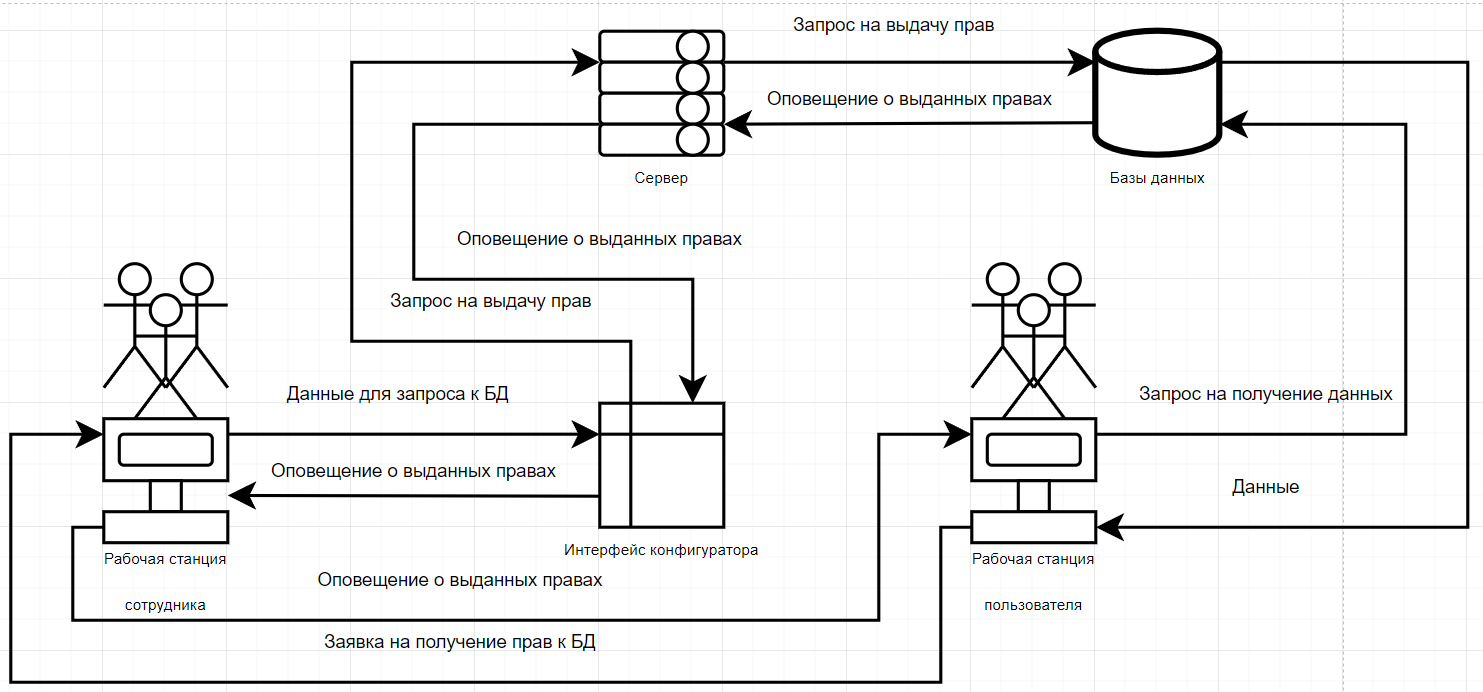
Взаимодействие элементов конфигуратора происходит следующим образом:

* Пользователь с рабочей станции обращается к сотруднику с запросом на назначение прав;
* Сотрудник обрабатывает запрос, решает какие права необходимо назначить и, с рабочей станции, вводит данные для запроса в пользовательский интерфейс;
* Пользовательский интерфейс формирует и отправляет запрос серверу для обработки и передачи в бд;
* Сервер производит манипуляции с базой данных, хранящейся на сервере базы данных;
* Сервер формирует результаты, передает их пользовательскому интерфейсу;
* Пользовательский интерфейс обрабатывает полученный результат и выдает сообщение об ошибке или успехе операции;
* Сотрудник сообщает пользователю об успешном назначении прав;
* Пользователь обращатся к ресурсам, к которым он получил права.

Данные действия повторяются каждый раз, когда пользователь обращается к сотруднику.

На рисунке 3.9 представлена структура информационных потоков конфигуратора.

|  |
| --- |
| Рисунок 3.9 — Структура информационных потоков конфигуратора |



# 3.2 Разработка функциональной схемы работы конфигуратора

В соответствии с техническим заданием должны быть реализованы следующие функции:

* Функция «Авторизация»:

1. внесение авторизационных данных (логина и пароля) для получения доступа к функционалу конфигуратора;

* Функция «Назначение прав в Postgres»:

1. создание пользователя в базе данных;
2. удаление пользователя в базе данных;
3. назначение пользователю прав на просмотр, изменение и удаление данных в базе данных;
4. отзыв у пользователя прав на просмотр, изменение и удаление данных в базе данных.

На рисунке 3.10 представлена функциональная схема конфигуратора управления правами пользователей.

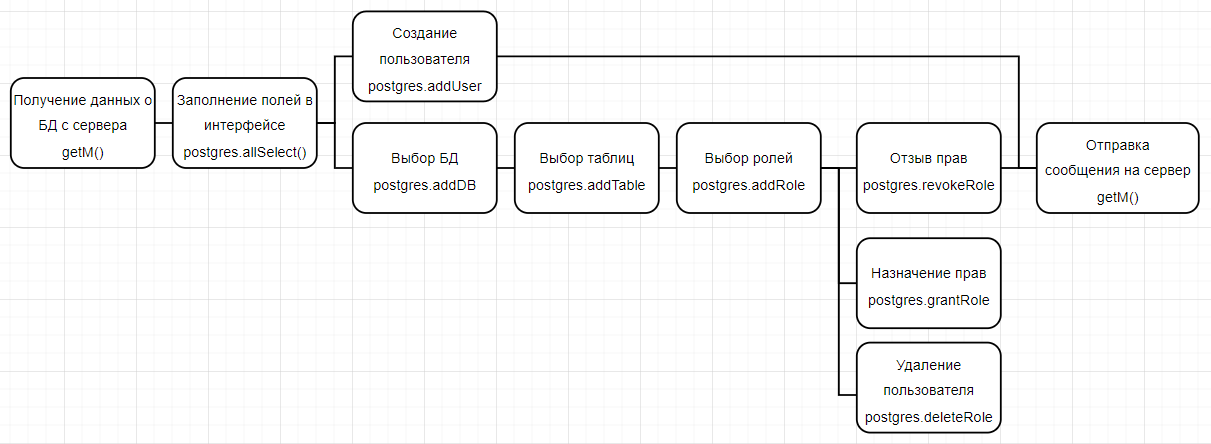


Рисунок 3.10 — Функциональная схема конфигуратора управления правами пользователей

Функция postgres.allSelect() отвечает за заполнение HTML форм данными полученными с помощью функции getM(), которая принимает сообщение сервера с данными о БД.

Функция postgres.addUser отвечает за создание пользователя. Она считывает значения логина и пароля из HTML формы, кодирует пароль, записывает их в сообщение и вызывает функцию postM(), отправляющую сообщение на сервер.

Функция postgres.addDB отвечает за выбор БД для назначения и отзыва прав. Она формирует список БД выбранных пользователем и добавляет их в сообщение, посылаемое на сервер.

Функция postgres.addTable отвечает за выбор таблиц для назначения и отзыва прав. Она формирует список таблиц выбранных пользователем и добавляет их в сообщение, посылаемое на сервер.

Функция postgres.addRole отвечает за выбор ролей для назначения и отзыва прав. Она формирует список ролей выбранных пользователем и добавляет их в сообщение, посылаемое на сервер.

Функция postgres.grantRole отвечает за назначение прав пользователю. Она определяет метод запроса и вызывает функцию postM, отправляющую сообщение на сервер.

Функция postgres.revokeRole отвечает за отзыв прав пользователю. Она определяет метод запроса и вызывает функцию postM, отправляющую сообщение на сервер.

Функция postgres.deleteRole отвечает за удаление пользователя. считывает значения логина пользователя из HTML формы, записывает его в сообщение и вызывает функцию postM(), отправляющую сообщение на сервер.

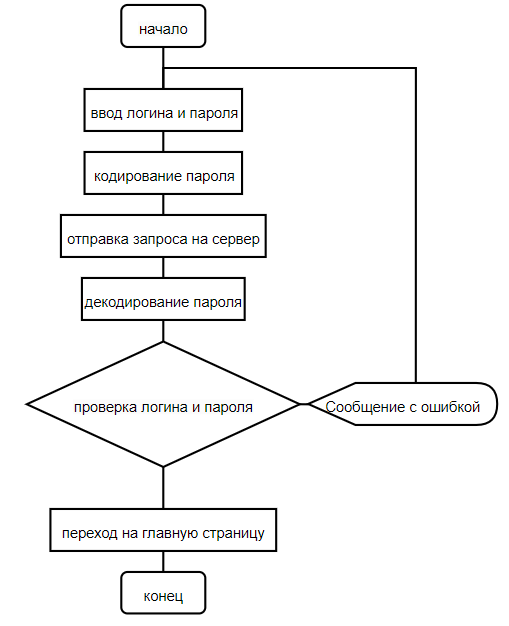
Функция getM() принимает сообщение с сервера.

Функция postM() отправляет сообщение на сервер.

# 3.3 Разработка алгоритмов работы конфигуратора

**Алгоритм авторизации**

Начало работы – открытие web-страницы авторизации. При введении существующих логина и пароля, происходит кодирование пароля на стороне клиента, отправка запроса на сервер, декодирование пароля на стороне сервера, авторизация при корректных значениях логина и пароля и переход на окно главной страницы. Для кодировки пароля используется функция CryptoJS.AES.encrypt("Message", "key"), а для декодировки функция CryptoJS.AES.decrypt(("Message", "key"), где Message это сообщение, которое необходимо закодировать, а key это ключ слова для шифрования и дешифрования.

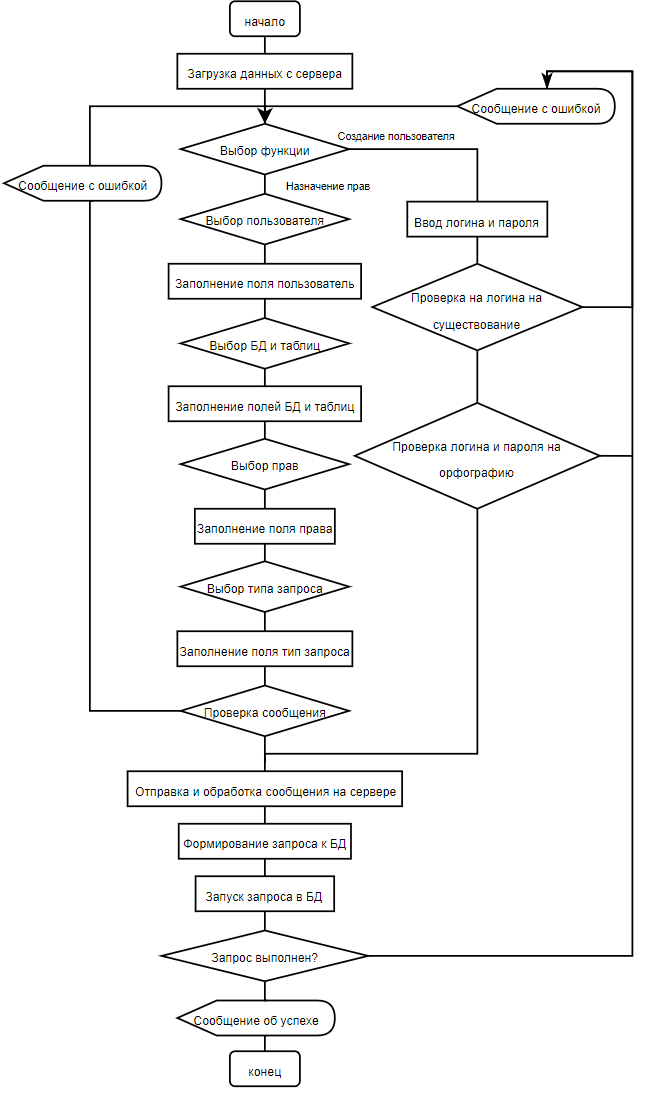


|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ЭВМ-201002.ВКР.1209.22.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Тропин М. Д. |  |  | Алгоритм авторизации | Литера | | | Лист | Листов |
| Пров | | Терентьев А. И. |  |  |  |  |  | 29 | 1 |
|  | |  |  |  | МГТУ ГА ФПМ и ВТ  ЭВМ 4-1 090301 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Феоктистова О.Г. |  |  |

**Алгоритм назначения прав**

При выборе назначения прав в Postgres происходит переход в окно назначения прав в Postgres где пользователь, после выбора стенда может совершить четыре действия: создать пользователя, введя логин и пароль, удалить выбранного пользователя, назначить и отозвать права пользователя, выбрав пользователя, базы данных, их таблицы и права пользователя, которые нужно назначить или отозвать. Список пользователей, баз данных, таблиц, прав и стендом формируется во время загрузки данных с сервера. При нажатии кнопки запуска любого из действий, описанных выше клиентская часть ПО сформирует сообщение, содержащие наименования всех введенных полей, заполнения всех выбранных флагов, название выбранного действия и отправит его на серверную часть ПО, где, исходя из данных в сообщении, будет сформированы SQL запросы ко всем выбранным БД.

При успешном выполнении запросов к БД, на экран будет выведено сообщение об успехе операции, при невыполнении запроса будет выведена ошибка.



|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ЭВМ-201002.ВКР.1209.22.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Тропин М. Д. |  |  | Алгоритм назначения прав | Литера | | | Лист | Листов |
| Пров | | Терентьев А. И. |  |  |  |  |  | 31 | 1 |
|  | |  |  |  | МГТУ ГА ФПМ и ВТ  ЭВМ 4-1 090301 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Феоктистова О.Г. |  |  |

# 3.4 Разработка структуры веб-сайта конфигуратора

Карта веб-сайта конфигуратора наглядно отображает каждую страницу, с которой будет взаимодействовать пользователь.

На рисунке 3.11 представлена карта веб-сайта конфигуратора.

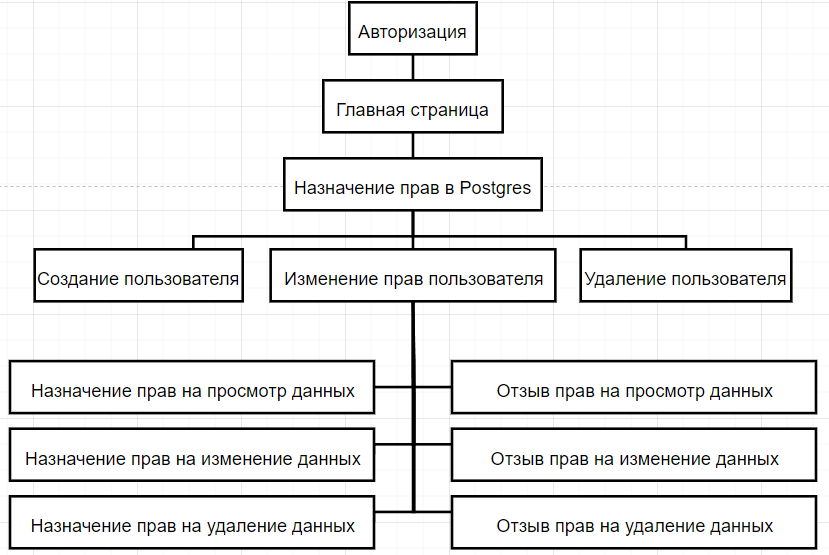


Рисунок 3.11 — Структура веб-сайта конфигуратора

Сайт состоит из 3 страниц:

1. Авторизация. Здесь сотрудник должен ввести свои данные для доступа к функционалу сайта.
2. Главная страница. Здесь сотрудник может посмотреть на свои персональные данные, и выбрать нужную ему страницу в меню.
3. Назначение прав в Postgres. Здесь сотрудник может создать пользователя, удалить пользователя или добавить пользователю права на просмотр, изменение и удаление данных в базе данных.

# 3.5 Разработка интерфейса веб-сайта конфигуратора

Пользовательский интерфейс — это интерфейс, с помощью которого человек может управлять программным обеспечением или аппаратным оснащением. Он должен быть удобными в использовании, чтобы взаимодействие с ними происходило на максимально интуитивном уровне.

При разработке пользовательского интерфейса необходимо придерживаться нескольким принципам его построения:

− Ясность. Интерфейс должен быть узнаваемым и назначение элементов должно быть очевидным для пользователя;

− Выделение главных элементов. Основные функции и элементы интерфейса должны быть броскими;

− Дизайн. Который должен позволить пользователям сконцентрироваться на решении задач, а не на работе интерфейса.

Для разработки интерфейса есть определённое название - верстка страниц. Вёрстка - представляет собой описание программными средствами внешнего вида страниц. Независимо от браузера и разрешения экрана, используемого устройства, страница должна отображаться верно - то есть должна присутствовать адаптивность.

Внешний вид страниц определяется html-тегами, они являются некими контейнерами, которые определяют, чем будет наполняться страница. Исходя из анализа технического задания, лучшем вариантом будет создать макет шаблон для всех страниц конфигуратора.

На рисунке 3.12 представлен макет страниц.

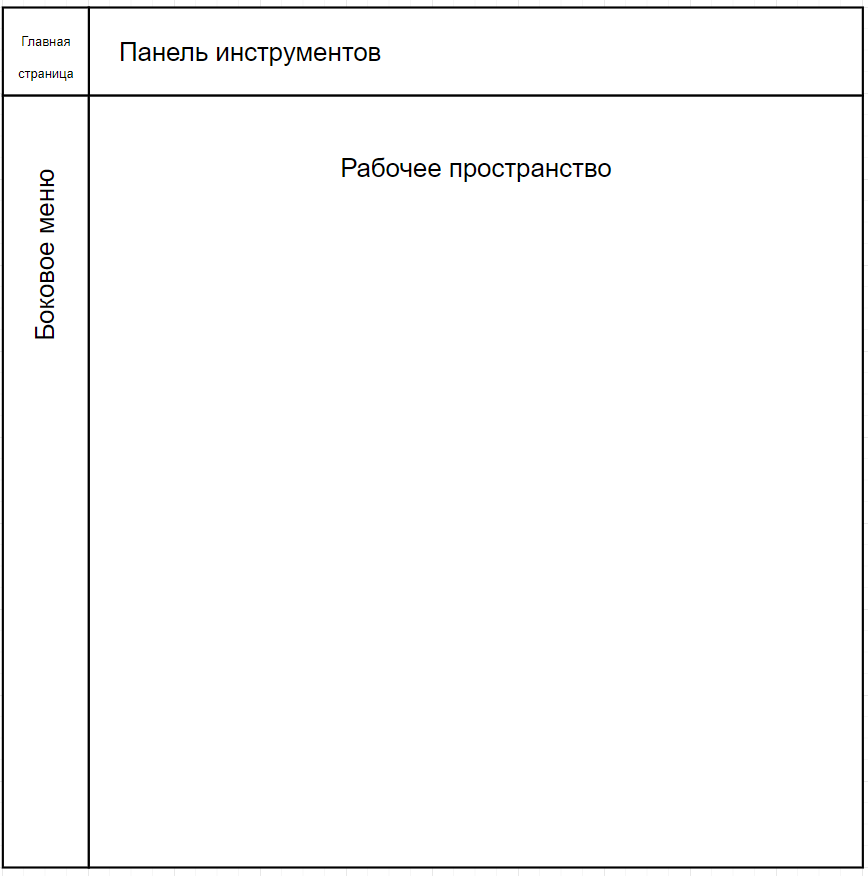


Рисунок 3.12 — макет страниц

Также, придерживаясь принципов, описанных ранее, в графическом редакторе были разработаны специальные картинки для кнопок в боковом меню.

На рисунке 3.13 представлена кнопка главного раздела.

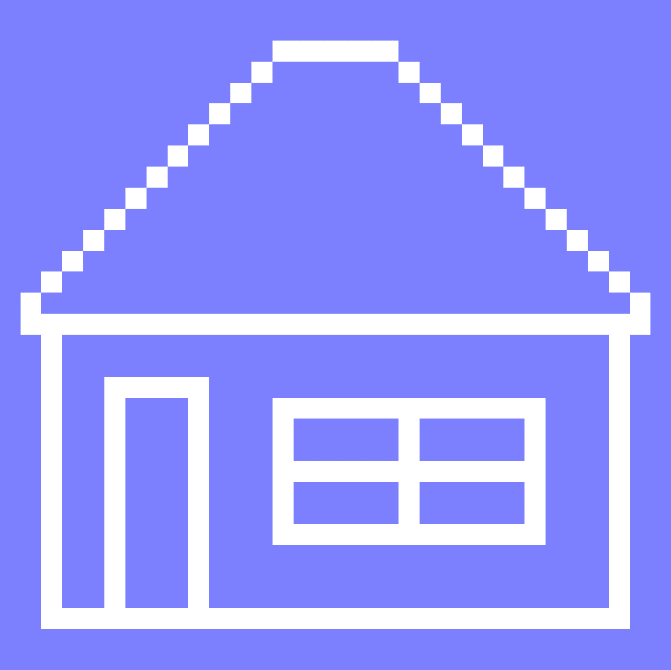


Рисунок 3.13 — кнопка главного раздела

На рисунке 3.14 представлена кнопка раздела Postgres.

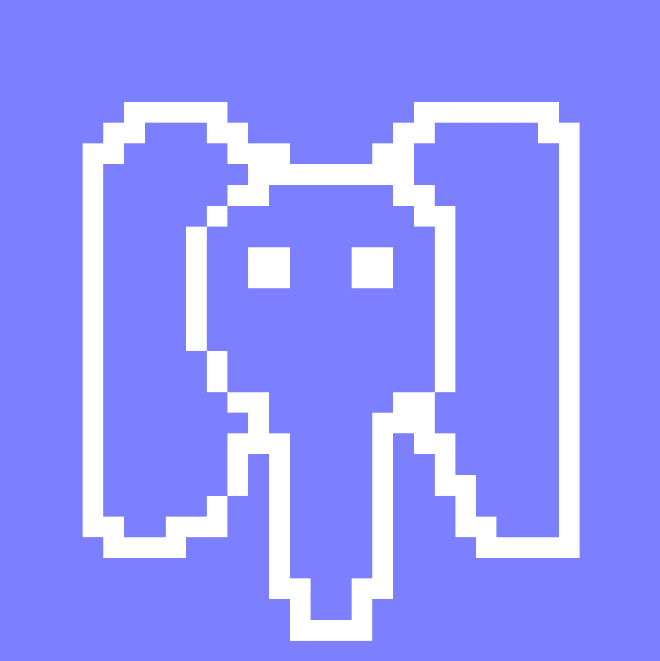


Рисунок 3.14 — кнопка раздела Postgres

# ЗАКЛЮЧЕНИЕ

В процессе выполнения выпускной квалификационной работы были выполнены следующие задачи:

* Показана актуальность создания конфигуратора управления правами пользователей, реализуемого для РНИС;
* Проведен анализ предметной области с определением достоинств недостатков избранных решений для создания конфигуратора управления правами пользователей, реализуемого для РНИС;
* Для создания конфигуратора выбраны: язык программирования JS, язык гипертекстовой рассылки HTML, программная платформа Node JS, среда разработки Visual Code и СУБД PostgresSQL;
* Разработаны структурная схема конфигуратора и схема информационных потоков конфигуратора.
* Разработаны алгоритмы работы конфигуратора.

Итогом проведенной работы является разработанный конфигуратор управления правами пользователей, который качественно упрощает и оптимизирует работу сотрудников РНИС. Конфигуратор выполнен в соответствии с техническим заданием. Поставленные цели достигнуты.

# СПИСОК ЛИТЕРАТУРЫ

1. Материалы веб–сайта «HTML» [Электронный ресурс], – Режим доступа: https://html.spec.whatwg.org/multipage/ свободный. (Дата обращения 20.04.2024 г.).

2. Материалы веб–сайта «wikipedia» [Электронный ресурс], – Режим доступа: https://ru.wikipedia.org/wiki/GitLab свободный. (Дата обращения 10.04.2024 г.).

3. Материалы веб–сайта «Современный учебник JavaScript» [Электронный ресурс], – Режим доступа: https://learn.javascript.ru/ свободный. (Дата обращения 17.04.2024 г.).

4. Материалы веб–сайта «nodejs» [Электронный ресурс], – Режим доступа: https://nodejs.org/en свободный. (Дата обращения 17.04.2024 г.).

# ПРИЛОЖЕНИЕ А

Спецификация

|  |  |  |
| --- | --- | --- |
| Обозначение | Наименование | Примечание |
|  | Документация |  |
|  |  |  |
| ЭВМ-201002.ВКР.1209.22.ПЗ | Конфигуратор управления |  |
|  | правами доступа пользователей |  |
|  | региональной |  |
|  | навигационно-информационной |  |
|  | системы |  |
|  | Текст программы |  |
|  |  |  |
| ЭВМ-201002.ВКР.1209.22.ПЗ | Конфигуратор управления |  |
|  | правами доступа пользователей |  |
|  | региональной |  |
|  | навигационно-информационной |  |
|  | системы |  |
|  | Руководство пользователя |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# ПРИЛОЖЕНИЕ Б

Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы

ТЕКСТ ПРОГРАММЫ

Листов 15

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ЭВМ-201002.ВКР.1209.22.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Тропин М. Д. |  |  | Приложение | Литера | | | Лист | Листов |
| Пров | | Терентьев А. И. |  |  |  |  |  | 39 | 1 |
|  | |  |  |  | МГТУ ГА ФПМ и ВТ  ЭВМ 4-1 090301 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Феоктистова О.Г. |  |  |

**Server.js**

const express = require('express')

const fs = require('fs')

const path = require('path')

const Pool = require('pg').Pool

var p = {

user: "postgres",

password: "postgres",

host: "localhost",

port: 5432,

database: "testDB1"

}

const pool = new Pool(p)

const app = express()

const PORT = 3000

const createPath = (page) => path.resolve(\_\_dirname, 'public', `${page}.html`)

const authJson = fs.readFileSync('public/scripts/json/auth.json', {encoding : 'utf8'})

const authPostgres = fs.readFileSync('public/scripts/json/postgres.json', {encoding : 'utf8'})

var authData = JSON.parse(authJson)

var postgresData = JSON.parse(authPostgres)

var num = false

var arr = []

app.use(express.static('public'))

app.use(express.json());

app.listen(PORT, () => {console.log(`Express web app on http://localhost:${PORT}`);})

app.get('/index', (req,res)=>{

res.send(num)

res.sendFile(createPath('index'))

})

app.post('/index', (req,res)=>{

const { parcel } = req.body

parcel.password = CryptoJS.AES.encrypt(parcel.password, "key")

num = false

for(var i=0;i<authData.length;i++){

if(authData[i].login == parcel.login && authData[i].password == parcel.password){

num = true

}

}

})

app.get('/main', (req,res)=>{

res.sendFile(createPath('main'))

})

app.get('/postgres', async (req,res)=>{

const GetRoles = await pool.query(`SELECT rolname FROM pg\_catalog.pg\_roles where rolcanlogin = 'true'`)

for(var i=0;i<GetRoles.rows.length;i++){

postgresData.user[i] = GetRoles.rows[i].rolname

}

const GetDbNames = await pool.query(`SELECT datname FROM pg\_catalog.pg\_database where datistemplate = 'false'`)

for(var i=0;i<GetDbNames.rows.length;i++){

postgresData.db[i] = GetDbNames.rows[i].datname

p.database = GetDbNames.rows[i].datname

arr[i] = new Pool(p)

const GetTNames = await arr[i].query(`SELECT tablename FROM pg\_catalog.pg\_tables WHERE schemaname = 'public';`)

for(var j=0;j<GetTNames.rows.length;j++){

postgresData.table.push(GetTNames.rows[j].tablename)

}

}

res.json(postgresData)

res.sendFile(createPath('postgres'))

})

app.post('/postgres', async (req,res)=>{

const { parcel } = req.body

console.log(parcel)

switch (parcel.requestType){

case 'add':

a = String(parcel.login)

b = String(parcel.password)

c = parcel.db

z = 'CREATE ROLE ' + a + ' WITH LOGIN ENCRYPTED PASSWORD ' + "'" + b + "'"

const addU = await pool.query(z)

res.json(addU)

break

case 'grant':

if (parcel.allRoleChecked == true){

a = 'ALL PRIVILEGES'

} else {

a = String(parcel.role)

}

b = parcel.db

d = String(parcel.user)

c = []

for(var i=0;i<b.length;i++){

p.database = b[i]

arr[i] = new Pool(p)

if (parcel.allTableChecked == true){

c = 'ALL TABLES IN SCHEMA public'

z = 'GRANT ' + a + ' ON ' + c + ' TO ' + d

} else {

const GetTNames = await arr[i].query(`SELECT tablename FROM pg\_catalog.pg\_tables WHERE schemaname = 'public';`)

for(var j=0;j<GetTNames.rows.length;j++){

for(var l=0;l<parcel.table.length;l++){

if (parcel.table[l] == GetTNames.rows[j].tablename) {

c.push(String('public.' + '"' + parcel.table[l] + '"'))

}

}

}

z = 'GRANT ' + a + ' ON ' + c + ' TO ' + d

}

if (c != ''){

console.log(z)

const grantU = await arr[i].query(z)

c = []

if (i == (b.length-1)){

res.json(grantU)

}

}

}

break

case 'revoke':

if (parcel.allRoleChecked == true){

a = 'ALL PRIVILEGES'

} else {

a = String(parcel.role)

}

b = parcel.db

d = String(parcel.user)

c = []

for(var i=0;i<b.length;i++){

p.database = b[i]

arr[i] = new Pool(p)

if (parcel.allTableChecked == true){

c = 'ALL TABLES IN SCHEMA public'

z = 'REVOKE ' + a + ' ON ' + c + ' FROM ' + d

} else {

const GetTNames = await arr[i].query(`SELECT tablename FROM pg\_catalog.pg\_tables WHERE schemaname = 'public';`)

for(var j=0;j<GetTNames.rows.length;j++){

for(var l=0;l<parcel.table.length;l++){

if (parcel.table[l] == GetTNames.rows[j].tablename) {

c.push(String('public.' + '"' + parcel.table[l] + '"'))

}

}

}

z = 'REVOKE ' + a + ' ON ' + c + ' FROM ' + d

}

if (c != ''){

const revokeU = await pool.query(z)

c = []

if (i == (b.length-1)){

res.json(revokeU)

}

}

}

break

case 'delete':

a = parcel.user

//удаление связей

z = 'DROP USER IF EXISTS ' + a

const deleteU = await pool.query(z)

res.json(deleteU)

break

default: break

}

})

app.use((req,res)=>{

res.sendFile(createPath('error'))

})

**Auth.js**

const Url = 'http://localhost:3000/index'

var auth = {

login: "\*\*\*",

password: "\*\*\*",

authorization: function(){

log = document.getElementById("login");

pass = document.getElementById("password");

auth.login = log.value

auth.password = pass.value

async function getAuth(){

const res = await fetch(Url, {

method: 'GET'

})

const result = await res.text()

if (result == 'true') {

window.location="main.html"

}

else {

alert( "Амогус" )

}

}

async function postAuth(){

const res = await fetch(Url, {

method: 'POST',

headers: {

"Content-Type": 'application/json'

},

body: JSON.stringify({

parcel: auth

})

})

}

postAuth()

getAuth()

}

}

function init() {

var Voiti = document.getElementById("Войти");

Voiti.onclick = auth.authorization;

}

window.onload = init;

**Main.js**

var index = {}

function init() {

index.display(index)

}

window.onload = init;

**Postgres.js**

const Url = 'http://localhost:3000/postgres'

var postgres = {

standList: [],

userNameList: [],

DBList: [],

tableList: [],

roleList: [],

allDB: false,

currentDBList: [],

allTable: false,

currenttableList: [],

allRole: false,

currentroleList: [],

allSelect: function(){

for(var i = 0; i<postgres.standList.length; i++){

document.getElementById("stand").options[document.getElementById("stand").options.length] = new Option(postgres.standList[i], i);

}

for(var i = 0; i<postgres.userNameList.length; i++){

document.getElementById("user").options[document.getElementById("user").options.length] = new Option(postgres.userNameList[i], i);

}

for(var i = 0; i<postgres.DBList.length; i++){

document.getElementById("db").options[document.getElementById("db").options.length] = new Option(postgres.DBList[i], i);

}

for(var i = 0; i<postgres.tableList.length; i++){

document.getElementById("table").options[document.getElementById("table").options.length] = new Option(postgres.tableList[i], i);

}

for(var i = 0; i<postgres.roleList.length; i++){

document.getElementById("role").options[document.getElementById("role").options.length] = new Option(postgres.roleList[i], i);

}

},

addDB: function(){

if (document.getElementById("cb1").checked) {

postgres.currentDBList = postgres.DBList;

document.getElementById("picked1").innerHTML = postgres.DBList;

} else {

if (postgres.currentDBList.indexOf(document.getElementById("db").options[document.getElementById("db").selectedIndex].text) == -1){

postgres.currentDBList.push(document.getElementById("db").options[document.getElementById("db").selectedIndex].text);

document.getElementById("picked1").innerHTML = postgres.currentDBList;

} else document.getElementById("picked1").innerHTML = postgres.currentDBList;

}

postgres.refresh()

},

addTable: function(){

if (document.getElementById("cb2").checked) {

postgres.currenttableList = postgres.tableList;

document.getElementById("picked2").innerHTML = postgres.tableList;

}else {

if (postgres.currenttableList.indexOf(document.getElementById("table").options[document.getElementById("table").selectedIndex].text) == -1){

postgres.currenttableList.push(document.getElementById("table").options[document.getElementById("table").selectedIndex].text);

document.getElementById("picked2").innerHTML = postgres.currenttableList;

} else document.getElementById("picked2").innerHTML = postgres.currenttableList;

}

postgres.refresh()

},

addRole: function(){

if (document.getElementById("cb3").checked) {

postgres.currentroleList = postgres.roleList;

document.getElementById("picked3").innerHTML = postgres.roleList;

}else {

if (postgres.currentroleList.indexOf(document.getElementById("role").options[document.getElementById("role").selectedIndex].text) == -1){

postgres.currentroleList.push(document.getElementById("role").options[document.getElementById("role").selectedIndex].text);

document.getElementById("picked3").innerHTML = postgres.currentroleList;

} else document.getElementById("picked3").innerHTML = postgres.currentroleList;

}

postgres.refresh()

},

Clear: function(){

document.getElementById("login").value = "";

document.getElementById("password").value = "";

postgres.currentDBList = [];

postgres.currenttableList = [];

postgres.currentroleList = [];

document.getElementById("picked1").innerHTML = postgres.currentDBList;

document.getElementById("picked2").innerHTML = postgres.currenttableList;

document.getElementById("picked3").innerHTML = postgres.currentroleList;

document.getElementById("cb1").checked = false;

document.getElementById("cb2").checked = false;

document.getElementById("cb3").checked = false;

postgres.refresh()

},

addUser: function(){

postgres.refresh()

message.requestType = "add"

postM()

},

grantRole: function(){

postgres.refresh()

message.requestType = "grant"

if((document.getElementById("cb1").checked == false && document.getElementById("picked1").innerHTML == [])

|| (document.getElementById("cb2").checked == false && document.getElementById("picked2").innerHTML == [])

|| (document.getElementById("cb3").checked == false && document.getElementById("picked3").innerHTML == [])){

alert( "Данные не введены" )

} else {

postM()

}

},

revokeRole: function(){

postgres.refresh()

message.requestType = "revoke"

if((document.getElementById("cb1").checked == false && document.getElementById("picked1").innerHTML == [])

|| (document.getElementById("cb2").checked == false && document.getElementById("picked2").innerHTML == [])

|| (document.getElementById("cb3").checked == false && document.getElementById("picked3").innerHTML == [])){

alert( "Данные не введены" )

} else {

postM()

}

},

deleteRole: function(){

postgres.refresh()

message.requestType = "delete"

postM()

},

refresh: function(){

message.login = document.getElementById("login").value

message.password = CryptoJS.AES.encrypt(document.getElementById("password").value "key")

message.stand = document.getElementById("stand").options[document.getElementById("stand").selectedIndex].text

message.user = document.getElementById("user").options[document.getElementById("user").selectedIndex].text

message.allDbChecked = document.getElementById("cb1").checked

message.allTableChecked = document.getElementById("cb2").checked

message.allRoleChecked = document.getElementById("cb3").checked

message.db = postgres.currentDBList

message.table = postgres.currenttableList

message.role = postgres.currentroleList

},

}

async function getM(){

const res = await fetch(Url, {

method: 'GET'

})

const result = await res.json()

postgres.standList = result.stand

postgres.userNameList = result.user

postgres.DBList = result.db

postgres.tableList = result.table

postgres.roleList = result.role

}

async function postM(){

const res = await fetch(Url, {

method: 'POST',

headers: {

"Content-Type": 'application/json'

},

body: JSON.stringify({

parcel: message

})

})

const result = await res.json()

if (result.command) {

alert( "Запрос " + result.command + " выполнен" )

}

else {

alert( "Запрос невыполнен" )

}

}

var message = {

login: "",

password: "",

stand: "",

user: "",

allDbChecked: false,

allTableChecked: false,

allRoleChecked: false,

db: [''],

table: [''],

role: [''],

requestType: ""

}

function init() {

getM().then(() => {

postgres.allSelect()

})

var Create = document.getElementById("Создать")

Create.onclick = postgres.addUser

var Add1 = document.getElementById("add1")

Add1.onclick = postgres.addDB

var Add2 = document.getElementById("add2")

Add2.onclick = postgres.addTable

var Add3 = document.getElementById("add3")

Add3.onclick = postgres.addRole

var Clear = document.getElementById("Очистить")

Clear.onclick = postgres.Clear

var Grant = document.getElementById("Выдать")

Grant.onclick = postgres.grantRole

var Revoke = document.getElementById("Отозвать")

Revoke.onclick = postgres.revokeRole

var Delete = document.getElementById("Удалить")

Delete.onclick = postgres.deleteRole

}

window.onload = init;

**Index.html**

<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>Авторизация</title>

<link rel="stylesheet" type="text/css" href="css/style.css">

</head>

<body>

<div class="menubar">

</div>

<div class="head">

<p>Авторизация</p>

</div>

<div class="workspace">

<p>

<label for="log">Логин:</label>

<input type="text" name="log" id="login">

</p>

<p>

<label for="pass">Пароль:</label>

<input type="text" name="pass" id="password">

</p>

<p>

<input type="submit" value="Войти" id="Войти">

</p>

</div>

<script type="text/javascript" src="scripts/auth.js"></script>

</body>

</html>

**Main.html**

<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>Инструмент L3</title>

<link rel="stylesheet" type="text/css" href="css/style.css">

</head>

<body>

<div class="menubar">

<ul>

<li><a href="main.html"><img src="img/main.png" width="80"

height="80" alt="Главная" title="Главная"></a></li>

<li><a href="postgres.html"><img src="img/postgres.png" width="80"

height="80" alt="Выдача прав в Postgres" title="Выдача прав в Postgres"></a></li>

</ul>

</div>

<div class="head">

<p>Главная</p>

</div>

<div class="workspace">

<p>

<label for="log">Добро пожаловать в devtool</label>

</p>

<p>

<label for="pass">Выбирете раздел</label>

</p>

</div>

<script type="text/javascript" src="scripts/index.js"></script>

</body>

</html>

**Postgres.html**

<!doctype html>

<html>

<head>

<meta charset="UTF-8">

<title>Назначение прав</title>

<link rel="stylesheet" type="text/css" href="css/style.css">

</head>

<body>

<div class="menubar">

<ul>

<li><a href="main.html"><img src="img/main.png" width="80"

height="80" alt="Главная" title="Главная"></a></li>

<li><a href="postgres.html"><img src="img/postgres.png" width="80"

height="80" alt="Назначение прав в Postgres" title="Назначение прав в Postgres"></a></li>

</ul>

</div>

<div class="head">

<p>Назначение прав в Postgres</p>

</div>

<div class="workspace">

<p>

<label for="stand">Выберите стенд</label>

<select name="stand" id="stand"></select>

</p>

<p>Создание пользователя</p>

<p>

<label for="name">Введите логин пользователя</label>

<input type="text" name="log" id="login">

</p>

<p>

<label for="name">Введите пароль пользователя</label>

<input type="text" name="pass" id="password">

</p>

<p>

<input type="submit" value="Создать" id="Создать">

</p>

<p>Назначение прав</p>

<p>

<label for="user">Выберите пользователя</label>

<select name="user" id="user"></select>

</p>

<p>

<label for="">Выдать права для всех БД на стенде?</label>

<input type="checkbox" name="db" value="1" id="cb1">

</p>

<p>

<label for="db">Выберите БД</label>

<select name="db" id="db"></select>

<label>Выбрано: </label>

<label id="picked1"></label>

</p>

<p>

<input type="submit" id="add1" value="Добавить">

</p>

<p>

<label for="">Выдать права для всех таблиц в БД?</label>

<input type="checkbox" name="table" value="2" id="cb2">

</p>

<p>

<label for="role">Выберите таблицы</label>

<select name="table" id="table"></select>

<label>Выбрано: </label>

<label id="picked2"></label>

</p>

<p>

<input type="submit" id="add2" value="Добавить">

</p>

<p>

<label for="">Выберите все права?</label>

<input type="checkbox" name="role" value="3" id="cb3">

</p>

<p>

<label for="role">Выберите права</label>

<select name="role" id="role"></select>

<label>Выбрано: </label>

<label id="picked3"></label>

</p>

<p>

<input type="submit" id="add3" value="Добавить">

</p>

<p>

<input type="submit" id="Очистить" value="Очистить">

</p>

<p>

<input type="submit" value="Выдать" id="Выдать">

<input type="submit" value="Отозвать" id="Отозвать">

<input type="submit" value="Удалить" id="Удалить">

</p>

</div>

<script type="text/javascript" src="scripts/postgres.js"></script>

</body>

</html>

**Style.css**

.menubar {

padding: 0;

margin: 0;

width: 100px;

height: 100%;

position: fixed;

top: 0;

left: 0;

background-color: #7C80FF;

}

.head {

padding: 0;

margin-left: 100px;

width: 95%;

height: 100px;

position: fixed;

top: 0;

background-color: gray;

}

.workspace {

padding: 0;

margin-left: 100px;

margin-top: 130px;

width: 90%;

height: 90%;

background-color: white;

}

body{

padding: 0;

margin: 0;

}

.menubar a {

color: black;

font-weight: normal;

color: white;

text-decoration: none;

width: 100px;

height: 100px;

padding: 0;

margin: 0;

}

.menubar li:last-of-type a {

border-right: none;

}

li {

list-style-type: none;

margin: 10px;

}

ul {

padding: 0;

margin: 0;

}

.head P{

margin-top: 25px;

margin-left: 15px;

font-size: 2em;

color: white;

}

.workspace P{

margin-top: 30px;

margin-left: 30px;

font-size: 15px;

}

.workspace input[type="submit"]{

padding: 10px 20px;

font-size: 15px;

background: gray;

cursor: pointer;

border-radius: 5px;

color: white;

}

.workspace input[type="text"]{

border-radius:5px;

border: none;

background:gainsboro;

color: #7C80FF;

padding: 5px;

width: 300px;

}

.workspace input[type="text"]:focus{

background: gray;

color:#000;

}

input[name~="log"] {

margin-left: 8px;

}

label[for~="login"] {

margin-left: 8px;

}

**Auth.json**

[

{

"login": "admin",

"password": "admin"

},

{

"login": "tropinm",

"password": "tropinm"

}

]

**Postgres. json**

{

"login": "",

"password": "",

"allDbChecked": false,

"allTableChecked": false,

"allRoleChecked": false,

"stand": ["RNIS 1", "RNIS 2", "RNIS 3"],

"user": [],

"db": [],

"table": [],

"role": ["SELECT", "INSERT", "UPDATE", "DELETE"],

"requestType": ""

}

**Package. json**

{

"name": "tool",

"version": "1.0.0",

"main": "server.js",

"scripts": {

"dev": "nodemon server.js",

"test": "echo \"Error: no test specified\" && exit 1",

"start": "node server.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"description": "",

"dependencies": {

"express": "^4.19.2",

"nodemon": "^3.1.2",

"pg": "^8.12.0"

}

}

# ПРИЛОЖЕНИЕ В

Конфигуратор управления правами доступа пользователей региональной навигационно-информационной системы

РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Листов 7

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | *ЭВМ-201002.ВКР.1209.22.ПЗ* | | | | | |
|  |  |  |  |  |
| Изм. | Лист | № докум | Подпись | Дата |
| Разраб | | Тропин М. Д. |  |  | Приложение | Литера | | | Лист | Листов |
| Пров | | Терентьев А. И. |  |  |  |  |  | 55 | 1 |
|  | |  |  |  | МГТУ ГА ФПМ и ВТ  ЭВМ 4-1 090301 | | | | |
| Н. Контр. | |  |  |  |
| Утв | | Феоктистова О.Г. |  |  |

**АННОТАЦИЯ**

В программном документе представлено руководство пользователя, предназначенное для помощи в использовании конфигуратора.

В данном руководстве содержатся требования к программным и аппаратным средствам для успешного выполнения программы, а также описаны действия пользователя при работе с конфигуратором.

**СОДЕРЖАНИЕ**

1. НАЗНАЧЕНИЕ КОНФИГУРАТОРА

2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ

3. ВЫПОЛНЕНИЕ ПРОГРАММЫ

**1. НАЗНАЧЕНИЕ КОНФИГУРАТОРА**

* Создание пользователей в PostgresSQL;
* Назначение прав пользователям в PostgresSQL;
* Отзыв прав у пользователей в PostgresSQL;
* Удаление пользователей в PostgresSQL.

**2. УСЛОВИЯ ВЫПОЛНЕНИЯ ПРОГРАММЫ**

**2.1. Условия эксплуатации**

Так как разработка является программным продуктом, условия эксплуатации совпадают с нормальными климатическими условиями эксплуатации компьютера.

**2.2. Требования к видам обслуживания**

Не предъявляются.

**2.3. Требования к количеству и квалификации персонала**

Не предъявляются.

**2.4. Требования к составу и параметрам технических и программных средств**

Минимальные требования для работы программы:

* персональный компьютер с операционной системой одной из:
* Linux 2.6 и выше;
* MacOS X 10.6 и выше;
* Windows XP и выше;
* объем оперативной памяти 8 Гб;
* объем накопителя 256 Гб;
* дисплей с разрешением 800х480 пикселей;
* СУБД PostgresSQL.

**3. ВЫПОЛНЕНИЕ ПРОГРАММЫ**

При переходе по ссылке или при первом запуске пользователь будет перенаправлен на страницу авторизации. Страница авторизации показана на рисунке 3.1.1.

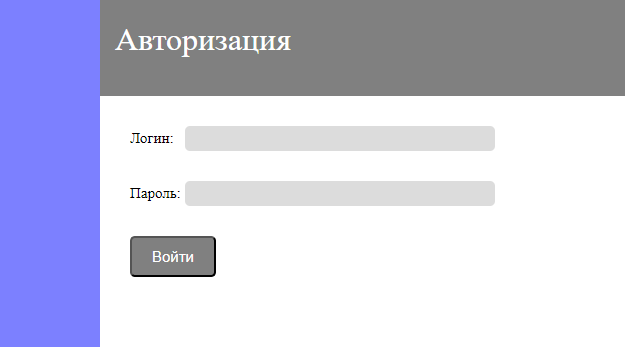


Рисунок 3.1.1 – Окно авторизации

На странице, пользователю будет необходимо ввести логин и пароль.

Если пользователь не имеет учётной записи, он должен обратится в отдел инфраструктуры компании. Нажав на кнопку войти, откроется окно главной страницы. Окно главной страницы показано на рисунке 3.1.2.

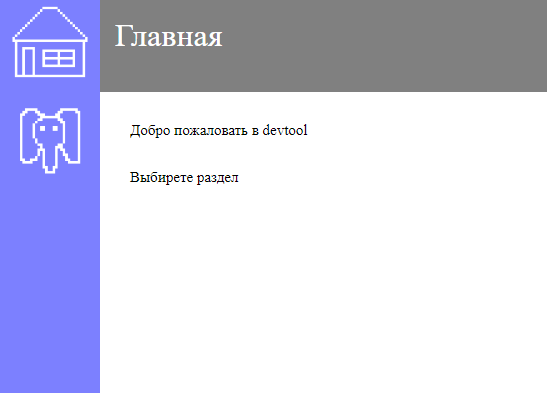


Рисунок 3.1.2 – Окно главной страницы

На главной странице пользователь может выбрать интересующий его раздел. На данный момент существует только раздел назначения прав в Postgres. Он ведет на окно назначения прав в Postgres. Окно назначения прав в Postgres показано на рисунке 3.1.3.

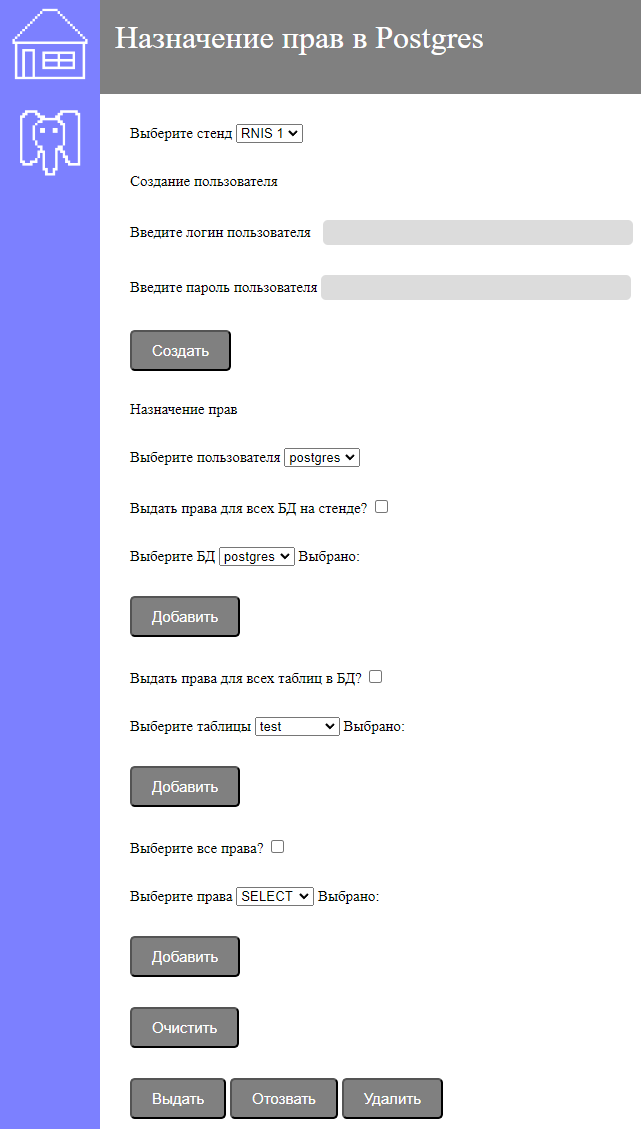
****

Рисунок 3.1.3 – Окно назначения прав в Postgres

На странице пользователь может выполнить ряд действий: создать пользователя БД, назначить права пользователю БД, отозвать права у пользователя БД, удалить пользователя БД.

Для создания пользователя БД нужно выбрать стенд, ввести логин и пароль в соответствующие поля и наджать кнопку создать.

Для назначения и отзыва прав пользователя БД нужно выбрать стенд, ввести логин пользователя, список БД, список таблиц и список прав в соответствующие поля и наджать кнопку выдать или отозвать.

Для удаления пользователя БД нужно выбрать стенд, ввести логин пользователя в соответствующее поле и наджать кнопку удалить.

Успешном выполнении запроса будет выведено окно с уведомлением. Окно с уведомлением показано на рисунке 3.1.4.

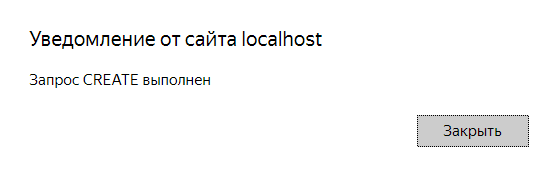


Рисунок 3.1.4 – Окно с уведомлением