

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**ОТЧЕТ**

по производственной практике

в ООО «Зрение 2.0», г. Ростов-на-Дону

Обучающийся Оганесьянц К.П.

подпись, дата И.О.Ф.

Обозначение отчета ПП.330000.000 Группа ВМО31

Направление 02.03.03 «Математическое администрирование информационных систем и технологий»

Профиль Программное обеспечение вычислительной техники и автоматизированных систем

Руководитель практики:

от предприятия Кудинов Н.Г.

должность подпись, дата И.О.Ф

М.П.

от кафедры Швидченко С.А.

должность подпись, дата И.О.Ф

Оценка

дата подпись преподавателя

Ростов-на-Дону

2022



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

|  |  |  |
| --- | --- | --- |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**ЗАДАНИЕ**

на производственную практику

вид практики

в ООО «Зрение 2.0», г. Ростов-на-Дону

в период с « 06 » июня 2022 г. по « 02 » июля 2022 г.

Обучающийся Оганесьянц Константин Павлович

Обозначение отчета ПП.330000.000 Группа ВМО31

Срок представления отчета на кафедру «04» июля 2022 г.

Содержание индивидуального задания:

Разработка базы данных и серверной архитектуры с элементами статистики для интернет-приложения стоматологии

Руководитель практики от

кафедры Швидченко С.А.

подпись, дата И.О.Ф

Задание принял к исполнению Оганесьянц К.П.

подпись, дата И.О.Ф



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| Зав. кафедрой | | «ПОВТиАС» |
| \_\_\_\_\_\_\_\_\_\_\_\_ | | В.В. Долгов |
| (подпись) | | (И.О.Ф.) |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. | |

**Рабочий график (план) проведения практики**

|  |  |  |
| --- | --- | --- |
| **№** | **Мероприятие** | **Срок выполнения** |
| 1 | Прохождение вводного и первичного инструктажа по охране труда на рабочем месте, и инструктажа по пожарной безопасности на объекте | 06.06.2022 |
| 2 | Получение индивидуального задания и постановка задачи | 07.06.2022 |
| 3 | Изучение теоретических материалов о задаче агрегации информации в стоматологической клиники | 08.06.2022 |
| 4 | Алгоритмическое конструирование | 09.06.2022 |
| 5 | Программная реализация | 12.06.2022 |
| 6 | Тестирование программного средства и подготовка итогового отчёта | 14.06.2022 |
| 8 | Защита итогового отчёта | 04.07.2022 |

Руководитель практики:

от предприятия Кудинов Н.Г.

должность подпись, дата И.О.Ф

М.П.

от кафедры Швидченко С.А.

должность подпись, дата И.О.Ф

Ростов-на-Дону

2022

ДНЕВНИК ПРОХОЖДЕНИЯ ПРАКТИКИ

|  |  |  |  |
| --- | --- | --- | --- |
| Дата | Место работы | Выполняемые работы | Оценка руководителя |
| 06.06.2022 | ООО «Зрение 2.0» | Ознакомление с территорией предприятия. Прохождение вводного и первичного инструктажа по охране труда на рабочем месте, и инструктажа по пожарной безопасности на объекте |  |
| 07.06.2022 | ООО «Зрение 2.0» | Получение индивидуального задания и постановка задачи |  |
| 08.06.2022 | ООО «Зрение 2.0» | Изучение теоретических материалов об арзитектуре баз данных и серверного ПО |  |
| с 09.06.2022  по 11.06.2022 | ООО «Зрение 2.0» | Алгоритмическое конструирование |  |
| с 12.06.2022  по 13.06.2022 | ООО «Зрение 2.0» | Программная реализация |  |
| с 14.06.2022  по 1.07.2022 | ООО «Зрение 2.0» | Тестирование программного средства и подготовка итогового отчета |  |
| 04.07.2022 | ООО «Зрение 2.0» | Защита итогового отчета |  |

Руководитель практики:

от предприятия Кудинов Н.Г.

должность подпись, дата И.О.Ф

М.П.

ОТЗЫВ - ХАРАКТЕРИСТИКА

Обучающийся Оганесьянц Константин Павлович

фамилия, имя, отчество

3 курса группы ВМО31 кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Вид практики производственная

Наименование места практики ООО «Зрение 2.0», г. Ростов-на-Дону

Обучающийся выполнил задания программы практики

Разработана архитектура базы данных и серверное программное средство с элементами статистики для веб-клиента стоматологического центра.

Дополнительно ознакомился/изучил

Во время разработки программного средства изучил особенности задачи построения баз данных. Ознакомился с функциями библиотеки fastify для применения технологии маршрутизации запросов.

Заслуживает оценки

|  |  |
| --- | --- |
|  | Руководитель практики  от предприятия  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кудинов Н.Г.  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_\_ г.  М.П. |

**Содержание**

Введение 7

1 Теоретический Обзор 8

2 Алгоритмическое конструирование 11

2.1 Алгоримт работы программного обеспечения и его схема 11

2.2 Вывод по главе 13

3 Программное конструирование 14

3.1 Выбор средств разработки 14

3.1.1 Выбор языка программирования 14

3.1.2 Выбор среды разработки 16

3.1.3 Выбор дополнительных инструментов 16

4.1 Проектирование виртуального лабораторного стенда 16

4.2 Проектирование функциональных модулей программы 17

4.3 Выводы по разделу 19

Заключение 22

Список использованных источников 23

Приложение А Техническое задание 24

Приложение Б Исходный код программного средства 30

### **Введение**

Современный мир невозможно представить без использования компьютерных технологий. Каждую компьютеризированную технологию можно разложить на две компоненты: алгоритмы и данные. Сегодня ежесекундно генерируются гигантские объемы данных, работать с подобными объемами данных без систем хранения и структурирования не предоставляется возможным. Для структурированной работы с полученной информацией необходимы инструменты, организующие работу, взаимодействие, хранение и доступ к данным. Таким инструментом и являются базы данных.

Вторым, необходимым в современных реалиях, инструментом работы с информацией являются информационные структуры, выраженные в веб-порталах и сайтах глобальной сети интернет

В силу данных сложностей была поставлена задача реализовать в рамках производственной практики программное обеспечение, реализующее возможность принимать, анализировать и хранить получаемую информацию в рамках работы стоматологической клиники а так же предоставлять по запросу статистический анализ заболеваний, с которыми чаще всего обращаются клиенты.

Следует отменить, что несмотря на то, что построение различных архитектур баз данных является, в большей степени, аналитической задачей в рамках проделанной работы удалось получить программное средство, выполняющее все необходимые действия для получения необходимых результатов.

В первом разделе рассматриваются общие представление о моделях и задачах баз данных. Производится анализ классических подходов к конструированию их архитектуры.

Во втором разделе рассматривается предметная область конечного продукта. Формируются изначальные представления моделей конечной базы данных.

Третий раздел характеризует тестирование прикладной программы (приложения) на задачах, которые последней необходимо будет выполнять в ходе работы.

#### Теоретический обзор

Приводится анализ предметной области, использования компьютерной техники а так же веб-ресурсов в работе современного предприятия.

#### Общее представление о построении и администрировании Баз данных

База данных — это информационная модель, позволяющая в упорядоченном виде хранить данные о группе объектов, обладающих одинаковым набором свойств [1]. Или, более узко: БД – именованная совокупность взаимосвязанных данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области, при такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений. БД состоит из множества связанных файлов и используется в целях отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Данные запоминаются так, чтобы они были независимы от программ, использующих эти данные. Для добавления новых или модификации существующих данных, а также для поиска данных в БД применяется система управления базами данных (СУБД).

СУБД – это программное обеспечение, предназначенное для создания, ведения и совместного использования БД многими пользователями. Простым языком, система управления базами данных выступает в качестве посредника между базой данных и ее пользователями [1]. СУБД осуществляют ввод, проверку, систематизацию, поиск и обработку данных, распечатку их в виде отчётов. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Первичным источником связанных данных при проектировании баз данных любых разновидностей является соответствующая предметная область, которая будет рассматриваться как совокупность знаний и данных об объектах и процессах, подлежащих проектированию и хранению в БД.

#### Анализ предметной области конечного продукта

**Предметной областью** данной курсовой работы является сфера персональных услуг, в процессе которых неизбежно образуется множество персональных данных, которые необходимо анализировать, сортировать и хранить для дальнейшего использования.

Пользователю, будучи работником стоматологической клиники достаточно просто взаимодействовать с системой. В процессе взаимодействия ведется учет клиентов, стоматологических заболеваний и записей.

Сперва рассмотрим предметную область на уровне представления.

База данных стоматологического центра необходима прежде всего для хранения информации о пользователях, заболеваниях, записях к специалисту.

Так как по принципам онлайн приложения конечным пользователем является сотрудник клиники, информация о пользователе должна содержать следующие данные: имя, фамилию, возраст а также необходимо иметь информацию о его записяъ. Данные в записи заполняет конечный пользователь при его создании.

Данные о записи должны содержать его дату, проблему, с которой клиент обратился, а так же информацию о процедурах.

#### Постановка задачи

Изучить теоретический материал о конструировании баз данных. Проанализировать противоречия между различными подходами. Исследовать различные подходы к реализации серверной архитектуры. Создать программное средство, отвечающее требованиям по хранению и учету персональных данных, появляющихся в процессе работы стоматологической клиники. Подробные требования к программному средству приведены в техническом задании (ПРИЛОЖЕНИЕ А).

#### 2.Алгоритмическое конструирование

В данной главе выполняется конструирование поведение сервера при различных действиях пользователя и приведена схема алгоритма, используемого сервером для шифрования данных

#### 2.1Алгоритм работы программного обеспечения и его схема

Серверное приложение представляет из себя сервис для получения и обработки запросов пользователя путем использования веб-клиента.

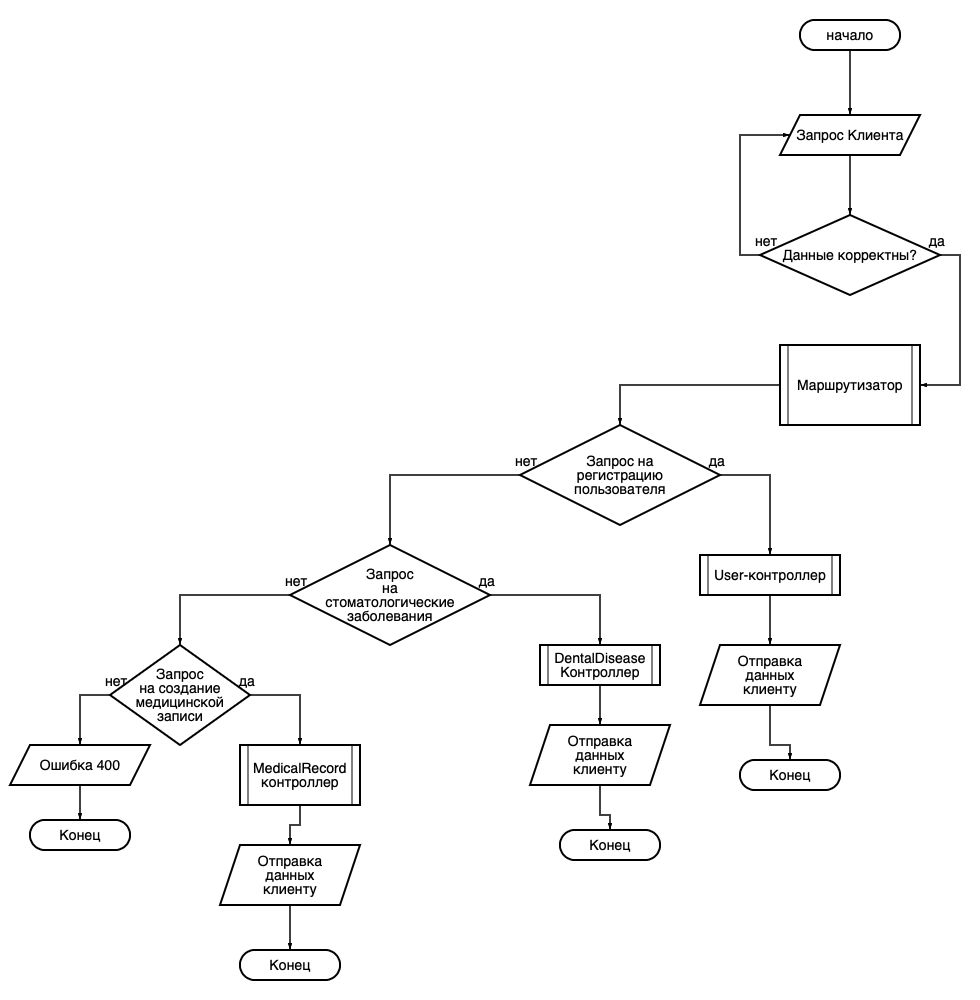
Описание работы сервера:

Самое первое действие, необходимое для коректного взаимодействия с системой это регистрация нового пользователя. Пользователь подает на вход запросом необходимые для регистрации данные: ФИО, возраст, логин и пароль. В случае, если пользователь с таким логином существует сервер выдаст предупреждение с просьбой изменить вводимые данные или войти в уже существующий аккаунт.

Дальнейшее взаимодействие пользователя с программой делится в зависимости от роли пользователя: клиент или стоматолог.

В случае стоматолога: клиент с такой ролью может создать дополнительные зубные заболевания и процедуры для них. Для такого создания ему необходимо отправить запросы на корректные выделенные адреса с коректными данными. В случае с зубными заболеваниями это будет только название, а при добавлении процедур так же дополнительно стоимость создаваемой услуги. Последним доступным действием для клиента-стоматолога будет создание медицинской записи, для которой необходимо указать заболевания, необходимые процедуры и дату.

Подробное взаимодействие с серверным продуктом изображены на блок-схеме 1.



Блок-схема 1 – алгоритмическая работа сервера

**2.2 Вывод по главе**

В данной главе был рассмотрен основной алгоритм маршрутизации при взаимодействии клиента с реализованным серверным приложением.

3 Программное конструирование

В данном разделе проводится обоснование выбора средств разработки программного средства, описание структуры программного средства и основных функций, которые реализуют необходимый функционал.

3.1 Выбор средств разработки

Выбор программного обеспечения обусловливается спецификой языка программирования, удобством использования, содержанием инструмента для решения сложных математических задач, взаимодействия с пользователем и осуществлением ввода-вывода, а также доступностью.

Практически все современные языки программирования в той или иной мере допускают использование различных парадигм. Парадигма программирования — это совокупность идей и понятий, определяющих стиль написания компьютерных программ (подход к программированию), это способ концептуализации, определяющий организацию вычислений и структурирование работы, выполняемой компьютером.

Парадигма программирования не определяется однозначно языком программирования. Так, на некоторых языках, которые не являются объектно-ориентированными, можно работать в соответствии с принципами объектно-ориентированного программирования, хотя это и сопряжено с определёнными сложностями, или функциональное программирование можно применять при работе на любом императивном языке, в котором имеются функции.

3.1.1 Выбор языка программирования

Практически все современные языки программирования имеют возможность подключения и работы с СУБД PostgreSQL. Поэтому следует определиться с выбором языка, рассмотри несколько из них, а именно: TypeScript и Python.

Python — высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Для удобной реализации перечисленных требований к интерфейсу БД «Онлайн склад» был выбран язык TypeScript. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), а также поддержку функционального подхода к программированию. В отличии от предыдущих языков, TypeScript отлично подходит для написания как фронтенд части приложения (с применением технологии React.js), так и серверной (при помощи рантайма Node.js).

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java и JavaScript — TypeScript, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, TypeScript в отличие от C++ не поддерживает множественное наследование классов и функциональное программирование[4].

3.1.2 Выбор среды разработки

Существует множество сред разработки подходящих для решения поставленной задачи, например, VS Code и WebStorm.

WebStorm поддерживает полный цикл разработки, прост в освоении, однако довольно перегружен функционалом, что приводит к замедлению разработки приложения.

VS Code является наиболее подходящей для решения поставленной задачи IDE, так как является бесплатной, официальной, обладает большим функционалом, имеет широкие возможности по разработке приложений, а также легковесен. Поэтому интерфейс для взаимодействия пользователей с базой данных был реализован на языке TypeScript в программной среде VS Code.

3.1.3 Выбор дополнительных инструментов

Для создания демонстрационного стенда, на котором конечное программное средство будет тестироваться необходимо выбрать технологию его конструирования.

Одной из самых современных технологий отображения интерфейсов в браузере на данный момент является React.js.

React — JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React разрабатывается и поддерживается Facebook, Instagram и сообществом отдельных разработчиков и корпораций. React может использоваться для разработки одностраничных и мобильных приложений.

**3.2 Проектирование виртуального лабораторного стенда**

Для тестирования реализованной серверной архитектуры необходимо создать программное средство.

Программное средство должно содержать модули, выполняющие взаимодействие пользователя с реализованным в рамках данной практике программным средством.

3.3 Проектирование функциональных модулей программы

Для организации взаимодействия пользователя с программным средством создан файл server, который описан в таблице 3.1.

Таблица 3.1 – Файл Server

|  |  |
| --- | --- |
| Функция | restServer |
| Описание функции | Обеспечивает создание и работу кластерного сервера |

Для реализации части маршрутизации реализованы функции, представленные ниже.

Для работы с моделью зубных заболеваний создана функция dentalDiseaeRoutes, которая описана в таблице 3.2.

Таблица 3.2 – Функция dentalDiseaeRoutes

|  |  |  |
| --- | --- | --- |
| Функция | dentalDiseaeRoutes | |
| Описание функции | задает маршрутизирующие пути работы с моделью стоматологических заболеваний | |
| Параметры | Тип параметра | Описание параметра |
| Fastify | FastifyInstance | Параметр, отвечающий за методы запросов |

Для работы с моделью процедур создана функция dentalProcedureRoutes, которая описана в таблице 3.3.

Таблица 3.3 – Функция dentalProcedureRoutes

|  |  |  |
| --- | --- | --- |
| Функция | dentalProcedureRoutes | |
| Описание функции | задает маршрутизирующие пути работы с моделью стоматологических процедур | |
| Параметры | Тип параметра | Описание параметра |
| Fastify | FastifyInstance | Параметр, отвечающий за методы запросов |

Для работы с моделью медицинских записей создана функция medicalRecordRoutes, которая описана в таблице 3.4.

Таблица 3.4 – Функция medicalRecordRoutes

|  |  |  |
| --- | --- | --- |
| Функция | medicalRecordRoutes | |
| Описание функции | задает маршрутизирующие пути работы с моделью медицинских записей | |
| Параметры | Тип параметра | Описание параметра |
| Fastify | FastifyInstance | Параметр, отвечающий за методы запросов |

Для организации взаимодействия серверного приложения с моделью зубных заболеваний в базе данных создан класс DentalDiseaseAPI. Класс и его методы описаны в таблице 3.5.

Таблица 3.5 – Класс DentalDiseaseAPI

|  |  |  |
| --- | --- | --- |
| Класс | DentalDiseaseAPI | |
| Описание класса | задает методы взаимодействия с моделью зубных заболеваний в базе данных | |
| Функции | Входящие параметры | Описание метода |
| CreateDentalDisease | name, token | Функция, отвечающая за создание записей о зубных заболевания в базе данных |
| getAllDentalDiseases |  | Функция позволяющая по запросу получить все записанные зубные заболевания |

Для организации взаимодействия серверного приложения с моделью зубных процедур в базе данных создан класс DentalProcedureAPI. Класс и его методы описаны в таблице 3.6.

Таблица 3.6 – Класс DentalProcedureAPI

|  |  |  |
| --- | --- | --- |
| Класс | DentalProcedureAPI | |
| Описание класса | задает методы взаимодействия с моделью зубных процедур в базе данных | |
| Функции | Входящие параметры | Описание метода |
| CreateDentalProcedure | name, token, price | Функция, отвечающая за создание записей о зубных процедурах в базе данных |
| getAllDentalProcedures |  | Функция позволяющая по запросу получить все записанные зубные процедуры |

Для организации взаимодействия серверного приложения с моделью медицинских записей в базе данных создан класс MedicalRecordAPI. Класс и его методы описаны в таблице 3.7.

Таблица 3.7 – Класс MedicalRecordAPI

|  |  |  |
| --- | --- | --- |
| Класс | MedicalRecordAPI | |
| Описание класса | задает методы взаимодействия с моделью медицинских записей в базе данных | |
| Функции | Входящие параметры | Описание метода |
| CreateOneMedicalRecord | token, userId, medicalDiseaseNames, medicalProcedureNames, dentistId | Функция, отвечающая за создание записей о медицинских записях в базе данных |
| getAllMedicalRecords |  | Функция позволяющая по запросу получить все зарегистрированные медицинские процедуры |

Листинг программы приведен в Приложении В.

3.4 Выводы по разделу

Проведено обоснование выбора средств разработки программного средства, описан процесс формирования структуры и архитектуры. Раскрывается разработка основных модулей.

1. **Тестирование отдельных модулей программного средства**

Для реализации выше разработанных алгоритмов и демонстрирования результатов был реализован веб-интерфейс, представляющий из себя админ-панель.

Для запуска необходимо открыть страницу админ-панели, которая отображена на рисунке 4.1.

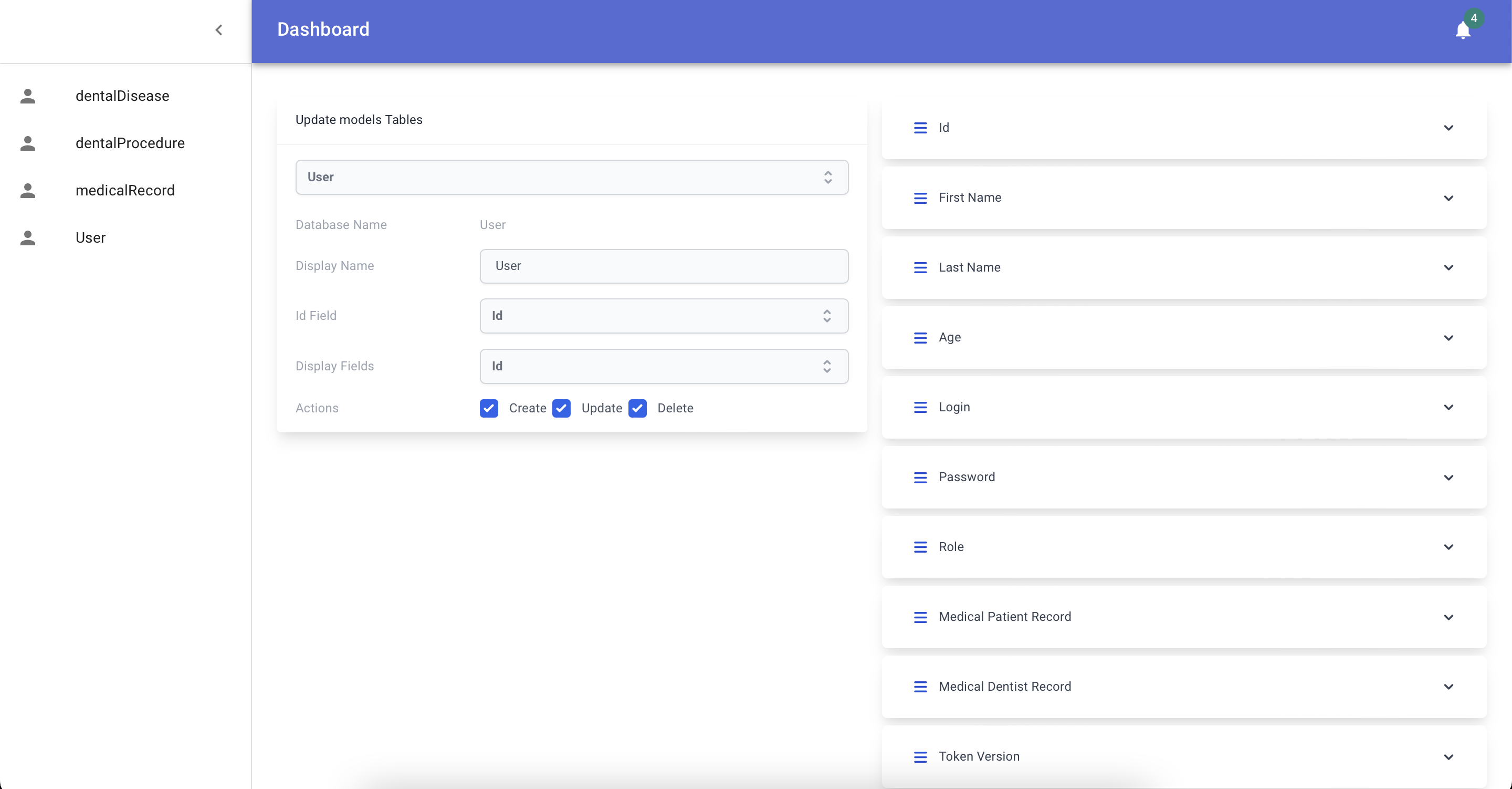


Рисунок 4.1 – Общий вид админ-панели

Для того чтобы проверить процесс регистрации и сохранение информации о пользователе необходимо в левом меню выбрать раздел “User”, как на рисунке 4.2.

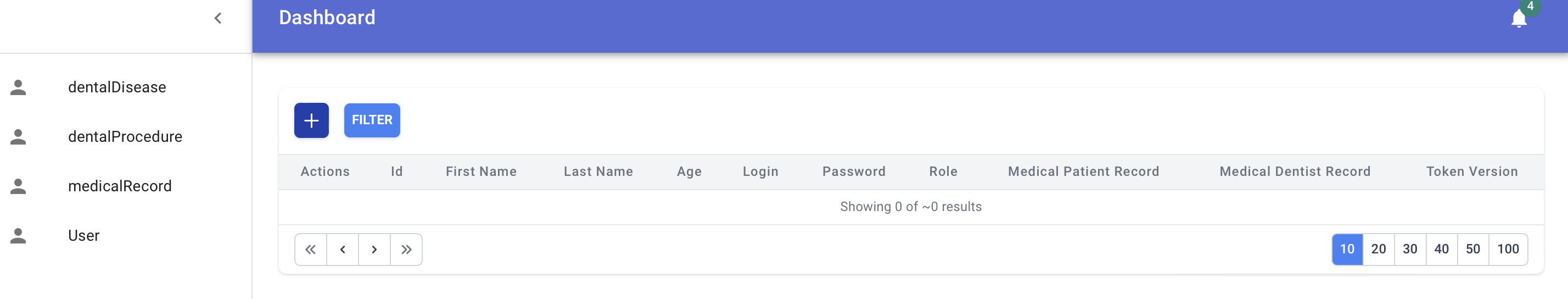


Рисунок 4.2 – Общий вид раздела “User”

Чтобы добавить новую запись в пользователей необходимо нажать на специализированную кнопку с иконкой «+», после чего откроется модальное окно создание пользователя, отображенное на рисунке 4.3.

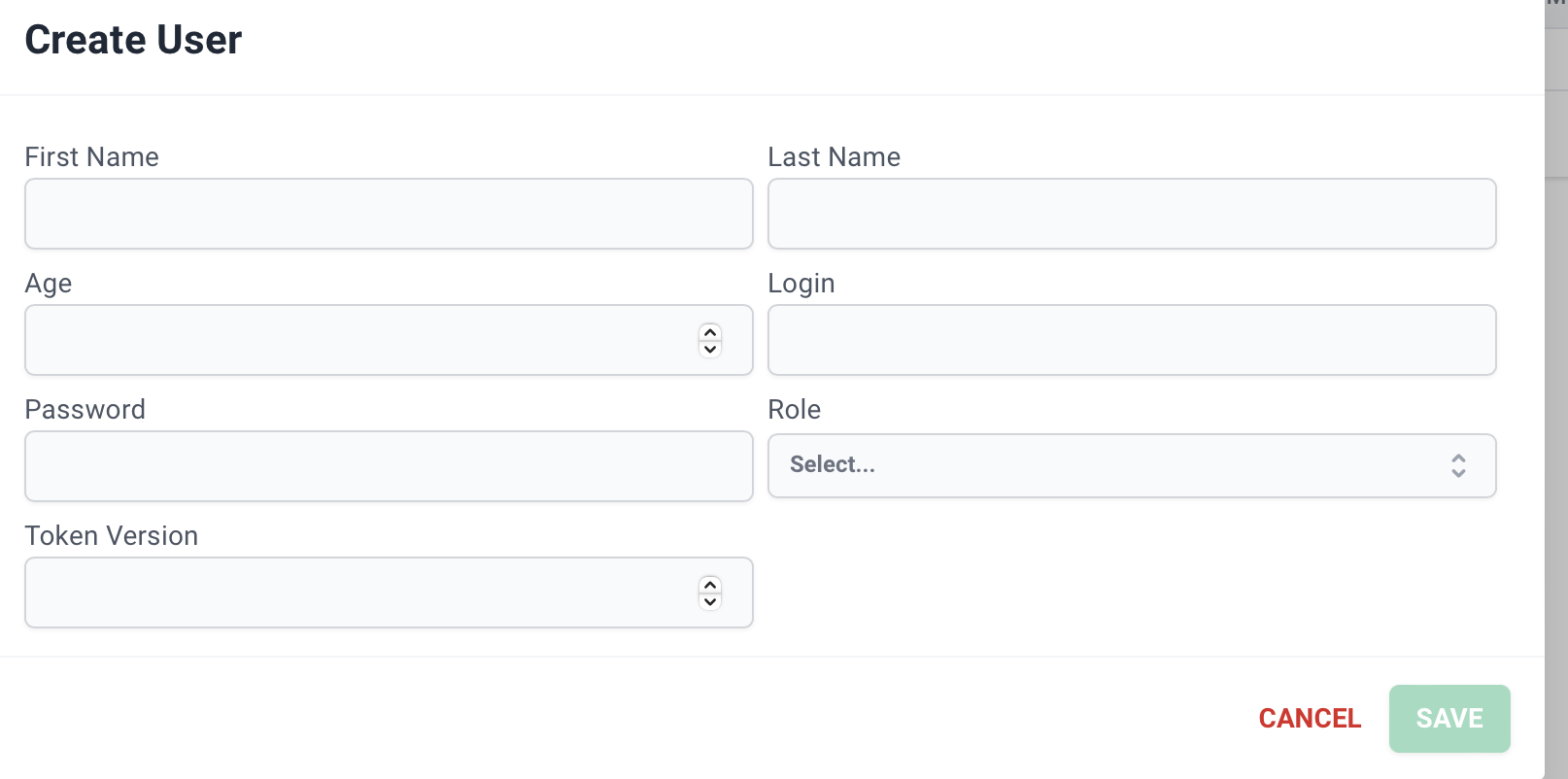


Рисунок 4.3 – Модальное окно создания записи пользователя

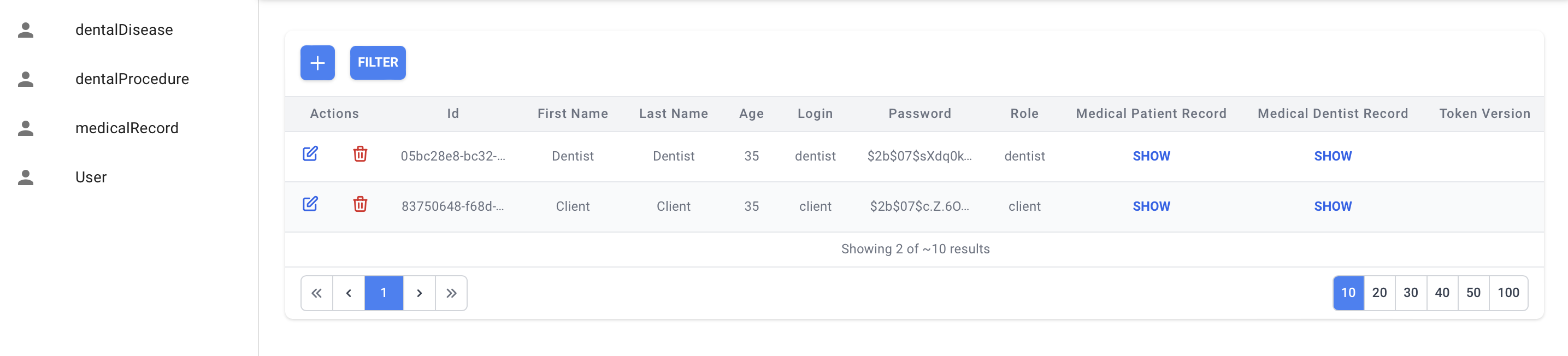


Рисунок 4.4 – Вид списка после создания тестовых пользователей

Взаимодействие с остальными моделями системы происходит по такому же сценарию. Для примера рассмотрим создание записи о стоматологическом заболевании.

Перейдем во вкладку этой модели и заполним модальное окно создания. Как видно на рисунке 4.4. для ввода доступны только необходимые для создания поля.

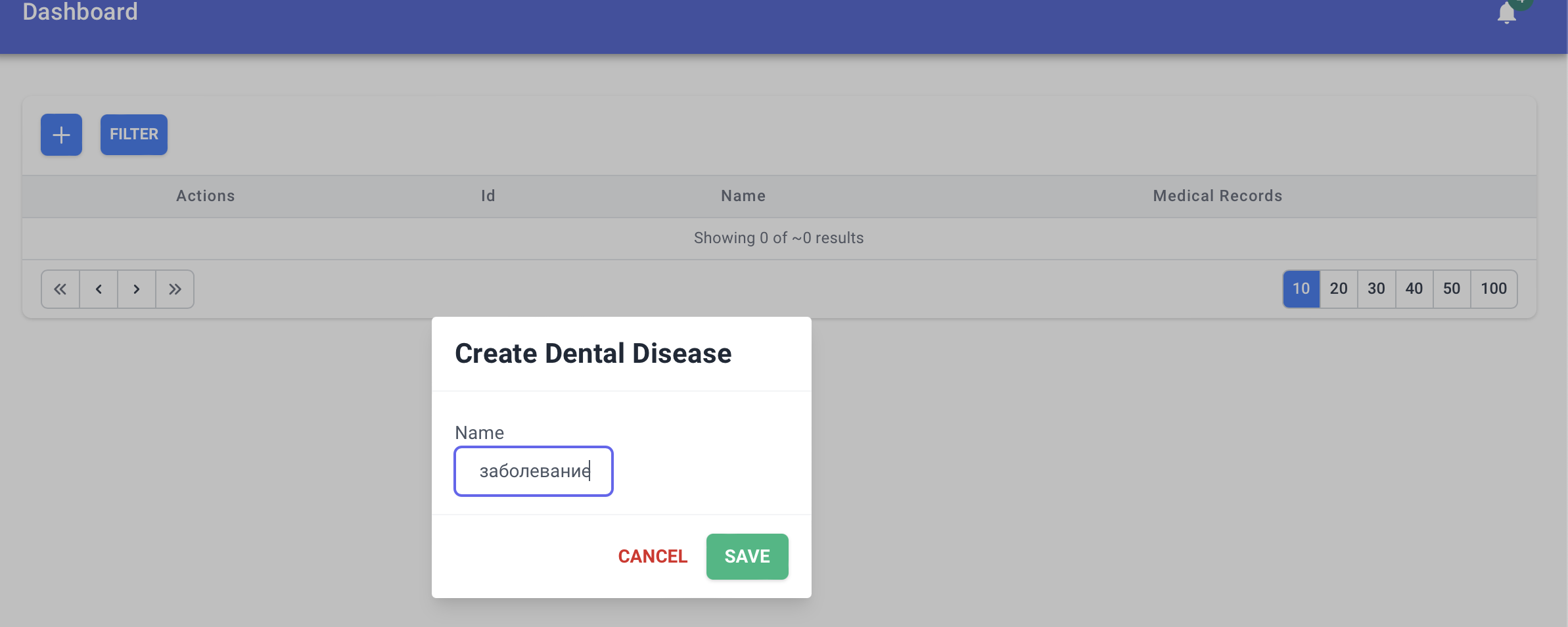


Рисунок 4.4 – вид списка после создания тестовых пользователей

### **Заключение**

Для решения поставленной задачи была изучена предметная область, определены конечные пользователи, описаны объекты и их атрибуты. В ходе выполнения данной работы была изучена выбранная СУБД PostgreSQL, ее возможности и функционал, а так же улучшены знания TypeScript, библиотеки React и Fastify.

.

### **Список использованных источников**

1. Дейт К.Дж. Введение в системы баз данных, 6-е издание: Пер. с англ. – К.; М.; СПб.: Издательский дом «Вильямс», 2000. – 848 с.
2. Конноли Т., Бегг К., Страчан А. Базы данных: проектирование, реализация и сопровождение. Теория и практика, 2-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2000. – 1120 с.
3. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.
4. Паттон Джефф. Пользовательские истории. Искусство гибкой разработки ПО. – СПб.: Питер, 2019.
5. Черный Борис Профессиональный TypeScript. Разработка масштабируемых JavaScript-приложений

### **Приложение А. Техническое задание**

|  |  |
| --- | --- |
| СОГЛАСОВАНО  Доцент, к.т.н.  каф. «ПОВТиАС»  \_\_\_\_\_\_\_\_\_\_\_/ Швидченко С.А.  «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022г. |  |

**А.1 Введение**

**А.1.1 Наименование программного средства**

Наименование программного средства – «База данных и серверная архитектура с элементами статистики для интернет-приложения стоматологии».

А.1.2 Область **применения**

Программное средство может применяться в сфере оказания персональных услуг в области стоматологии.

**А.2 Основание для разработки**

Разработка ведётся на основании документа «Учебный план для студентов ВУЗа» направление 02.03.02 «Математическое обеспечение и администрирование информационных систем» кафедры «Программное обеспечение вычислительной техники и автоматизированных систем» факультета «Информатика и вычислительная техника» Донского Государственного Технического Университета.

**А.3 Назначение разработки**

**А.3.1 Функциональное назначение**

Функциональное назначение программного средства заключается в создании системы маршрутизации для запросов при взаимодействии с веб-клиентом, а так же анализе и хранении персональных данных.

**А.3.2 Эксплуатационное назначение**

Эксплуатационное назначение состоит в использовании программного средства на выделенном сервере вместе с веб-клиентом.

**А.4 Требования к программе**

**А.4.1 Требования к функциональным характеристикам**

Программное средство должно осуществлять следующие функции:

* бесперебойное принятие запросов с веб-клиента;
* регистрация и хранение информации о пользователях;
* передача необходимой информации из базы данных по запросу;
* сохранение полученной в процессе взаимодействия с системой информации.

**А.4.2 Требования к надежности**

Для надежной работы программного средства необходимо, чтобы выполнялись следующие условия:

* бесперебойное питание технического средства, на котором находится продукт;
* регулярная проверка выделенного сервера на наличие вирусов;
* своевременное техническое обслуживание серверного оборудования;
* соблюдение техники безопасности при использовании технического средства.

**А.4.2.1 Входные данные**

В качестве входных данных используется набор персональных данных о пользователе:

* ФИО;
* Возраст;
* Логин;
* Пароль.

Входные данные для создания данных о стоматологических записях:

* название;
* id пациента
* id врача
* список необходимых процедур

Входные данные для создания данных о стоматологических заболеваниях:

* название;

**А.4.2.2 Выходные данные**

Выходные данные представлены в виде нескольких списков для каждого отдельного запроса, в которых указаны:

* необходимые для посещения клиенты;
* доступные стоматологические заболевания
* доступные медицинские процедуры;
* общий список стоматологических записей.

**А.4.3 Условия эксплуатации**

Для стабильного функционирования и оптимальной работы программного продукта необходимо соблюдение всех требований и правил эксплуатации вычислительной техники. Каких-либо требований к пользователю данного приложения нет.

**А.4.4 Требование к составу и параметрам технических средств**

В состав технических средств должен входить ПК с операционной системой Windows, Linux или MacOS, включающей в себя 64-разрядный процессор и оперативную память объемом не менее 4 Гбайт. Дополнительные требования и ограничения к составу и параметрам технических средств не вводятся.

**А.4.5 Требования к исходным кодам и языкам программирования**

Программное средство должно быть реализовано с применением языка программирования TypeScript(JavaScript), с использованием библиотек Fastify.

**А.4.6 Требования к упаковке и маркировке**

К упаковке и маркировке специальных требований не предъявляется.

**А.4.7 Требования к транспортировке и хранению**

Условия транспортирования, места хранения, условия складирования и сроки хранения в различных условиях должны соответствовать требованиям, предъявляемым к носителям информации, на которых будет содержаться данное ПО. Программное средство может храниться на любых цифровых носителях информации (жесткий диск, компакт-диск, флэш накопитель и т. п.).

**А.5 Требование к программной документации**

Программная документация состоит из следующего:

* титульный лист;
* лист задания на преддипломную практику;
* пояснительная записка к преддипломной практике;
* техническое задание по ГОСТ 19.201-78 ЕСПД;
* исходный код программного средства по ГОСТ 19.401-79 ЕСПД.

**А.6 Стадии и этапы разработки**

Реализация программного средства состояла из следующих этапов:

* постановка задачи (18.04.2022);
* анализ предметной области (19.04.2022 – 25.04.2022);
* алгоритмическое конструирование (26.04.2022 – 28.04.2022);
* программная реализация (29.04.2022 – 06.05.2022);
* тестирование программного средства (07.05.2022 – 13.05.20212;
* подготовка итогового отчета (07.05.2022 – 13.05.2022).

**А.7 Порядок и контроль приемки**

Порядок и контроль приемки определяются заведующим кафедрой «ПОВТиАС», подразумевающие собой демонстрацию показателя владения средствами для разработки программных средств в различных направлениях.

Главным требованием к приемке является наличие корректно работающего программного средства и отчета, предоставленного в печатном виде.

|  |  |
| --- | --- |
| Разработчик технического задания:  «20» июня 2022 г. | /Оганесьянц Константин Павлович/  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |

### **Приложение Б. Исходный код программного средства**

Листинг Б.1 – Программа «server.ts» (для языка JavaScript v16.13.0)

import fastify from 'fastify';

import fastifySwagger from '@fastify/swagger';

import { PORT, JWT\_SECRET } from './config';

import { logger } from './logger';

import baseRoute from './routes/base.routes';

import { userRoutes } from './routes/user.routes';

import { dentalDiseaeRoutes } from './routes/dentalDisease.routes';

import { dentalProcedureRoutes } from './routes/dentalProcedure.routes';

import { medicalRecordRoutes } from './routes/medicalRecord.routes';

const whitelist = [

'http://localhost:5000',

'http://localhost:4000',

'http://localhost:3000',

];

const server = fastify();

server.register(fastifySwagger, {

routePrefix: '/documentation',

swagger: {

info: {

title: 'Dental-clinit restAPI',

description: 'Documentation',

version: '1.0.0',

},

host: 'localhost',

schemes: ['https', 'http'],

consumes: ['application/json'],

produces: ['application/json'],

securityDefinitions: {

apiKey: {

type: 'apiKey',

name: 'apiKey',

in: 'header',

},

},

security: [{

apiKey: [],

}],

},

exposeRoute: true,

});

server.register(baseRoute);

server.register(userRoutes, { prefix: 'user' });

server.register(dentalDiseaeRoutes, { prefix: 'dentalDisease' });

server.register(dentalProcedureRoutes, { prefix: 'dentalProcedure' });

server.register(medicalRecordRoutes, { prefix: 'medicalRecord' });

const restServer = () => {

server.listen(PORT, '0.0.0.0', () => {

logger.info(`Server started at http://0.0.0.0:${PORT}`);

});

};

export {

restServer,

};

***Листинг Б.2 – Программа «MedicalRecord.ts » (для языка JavaScript v16.13.0)***

import { medicalRecord } from '@prisma/client';

import prisma from '../prisma-client';

import { verifyToken } from '../integrations/jwt';

class MedicalRecordAPI {

public static async createOneMedicalRecord ({ token, userId, medicalDiseaseNames, medicalProcedureNames, dentistId }: createOneMedicalRecordArgs): Promise<medicalRecord> {

const verifiedToken = await verifyToken(token);

if (!verifiedToken) {

throw new Error('Not authorized.');

}

const existingProcedures = await prisma.dentalProcedure.findMany({

where: {

name: {

in: medicalProcedureNames,

},

},

});

const existingDiseases = await prisma.dentalDisease.findMany({

where: {

name: {

in: medicalDiseaseNames,

},

},

});

if (!existingDiseases || !existingProcedures) {

throw new Error('Provided disease or procedure does not exist in current database.');

}

const totalPrice = existingProcedures.reduce((acc, procedure) => {

return acc + procedure.price;

}, 0);

const record = await prisma.medicalRecord.create({

data: {

patient: {

connect: {

id: userId,

},

},

procedures: {

connect: existingProcedures.map((procedure) => {

return { id: procedure.id };

}),

},

diseases: {

connect: existingDiseases.map((disease) => {

return { id: disease.id };

}),

},

dentist: {

connect: {

id: dentistId,

},

},

date: new Date(),

totalPrice,

},

});

if (!record) {

throw new Error('An error occured.');

}

return record;

}

public static async getAllMedicalRecords (token: string) {

const verifiedToken = await verifyToken(token);

if (!verifiedToken) {

throw new Error('Not authorized.');

}

return prisma.medicalRecord.findMany({

include: {

patient: true,

procedures: true,

dentist: true,

diseases: true,

},

});

}

}

interface createOneMedicalRecordArgs {

dentistId: string

userId: string

medicalProcedureNames: string[]

medicalDiseaseNames: string[]

token: string

};

export default MedicalRecordAPI;

***Листинг Б.3 – Программа «MedicalProcedure.ts » (для языка JavaScript v16.13.0)***

import { dentalProcedure } from '@prisma/client';

import { verifyToken } from '../integrations/jwt';

import prisma from '../prisma-client';

class DentalProcedureAPI {

public static async createDentalProcedure ({ name, token, price }: createDentalProcedureArgs): Promise<dentalProcedure> {

const existingProcedure = await prisma.dentalProcedure.findUnique({

where: {

name,

},

});

if (existingProcedure) {

throw new Error('Dental procedure with provided name already exist.');

}

const verifiedToken = await verifyToken(token);

if (!verifiedToken) {

throw new Error('Not authorized.');

}

const procedure = await prisma.dentalProcedure.create({

data: {

name,

price,

},

});

if (!procedure) {

throw new Error('An error occured.');

}

return procedure;

}

public static async getAllDentalDiseases (userToken: string): Promise<dentalProcedure[]> {

const verifiedToken = await verifyToken(userToken);

if (!verifiedToken) {

throw new Error('Not authorized!');

}

return prisma.dentalProcedure.findMany({});

}

};

interface createDentalProcedureArgs {

name: string

price: number

token: string

}

export default DentalProcedureAPI;

Листинг Б4 – Программа «MedicalDisease.ts » (для языка JavaScript v16.13.0)

import { dentalDisease } from '@prisma/client';

import { verifyToken } from '../integrations/jwt';

import prisma from '../prisma-client';

class DentalDiseaseAPI {

public static async CreateDentalDisease ({ name, token }: CreateDentalDiseaseArgs): Promise<dentalDisease> {

const existingDisease = await prisma.dentalDisease.findUnique({

where: {

name,

},

});

const verifiedToken = await verifyToken(token);

if (!verifiedToken) {

throw new Error('Not authorized!');

}

if (existingDisease) {

throw new Error('Disease with provided name already exists.');

}

const disease = await prisma.dentalDisease.create({

data: {

name,

},

});

if (!disease) {

throw new Error('An error occured');

}

return disease;

}

public static async getAllDentalDiseases (userToken: string): Promise<dentalDisease[]> {

const verifiedToken = await verifyToken(userToken);

if (!verifiedToken) {

throw new Error('Not authorized!');

}

return prisma.dentalDisease.findMany({});

}

}

interface CreateDentalDiseaseArgs {

name: string

token: string

};

export default DentalDiseaseAPI;

Листинг Б.5 – Программа «User.ts » (для языка JavaScript v16.13.0)

import { FastifyReply, FastifyRequest, FastifyInstance } from 'fastify';

import { userRole } from '@prisma/client';

import bcrypt from 'bcrypt';

import prisma from '../prisma-client';

import { HASH\_SALT } from '../config';

import { createToken } from '../integrations/jwt';

class UserController {

public static async signUp (req: FastifyRequest<{

Body: signUpArgs

}>, reply: FastifyReply) {

const existingUser = await prisma.user.findUnique({

where: {

login: req.body.login,

},

});

if (existingUser) {

throw new Error('User with provided login is already exist.');

}

const hashedPassword = await bcrypt.hash(req.body.password, HASH\_SALT);

const user = await prisma.user.create({

data: {

firstName: req.body.firstName,

lastName: req.body.lastName,

password: hashedPassword,

role: req.body.role,

login: req.body.login,

age: req.body.age,

},

});

const token = createToken(user);

reply.send({ token });

}

public static async signIn (req: FastifyRequest<{

Body: signInArgs

}>, reply: FastifyReply) {

const user = await prisma.user.findUnique({

where: {

login: req.body.login,

},

});

if (!user) {

throw new Error('Incorrect login or password');

}

const passwordMatch = await bcrypt.compare(req.body.password, user.password);

if (!passwordMatch) {

throw new Error('Incorrect password.');

}

const token = createToken(user);

reply.send({ token });

}

}

interface signUpArgs {

login: string,

firstName: string,

lastName: string,

password: string

role?: userRole

age: number

}

interface signInArgs {

login: string,

password: string

}

export { UserController };