Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В.Непомнящий
подпись инициалы, фамилия
«___» _____2023г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Backend API для приложения SensoRehab компании «Sensomed»
тема

Руководитель	_____	_____	Л. И. Покидышева
	подпись, дата	должность, ученая степень	инициалы, фамилия
Студент	_____		Т. О. Салтымаков
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	_____	Л. И. Покидышева
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2023

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____О.В.Непомнящий
подпись инициалы, фамилия
«___» _____2023г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Салтымакову Тимофею Олеговичу
фамилия, имя, отчество

Группа КИ19-06Б Направление (специальность) 090301
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы: Backend API для приложения SensoRehab компании «Sensomed»

Утверждена приказом по университету № 4765/С от 23.03.2023

Руководитель ВКР: Покидышева Л. И., канд. техн. наук, доцент кафедры ВТ
инициалы, фамилия, учёная степень, должность, место работы

ИКИТ СФУ

Исходные данные для ВКР:

1) Сформулированные для ВКР цели и задачи.

2) Существующие системы электронной коммерции конкурентов.

Перечень разделов ВКР:

1) Анализ технического задания

2) Этапы проектирования

3) Программная реализация приложения

Перечень графического материала: Презентация в формате Microsoft PowerPoint

Руководитель ВКР _____ Л.И. Покидышева
подпись инициалы, фамилия

Задание принял к исполнению _____ Т.О.Салтымаков
подпись инициалы, фамилия

«22» 12 2022 г.
дата

РЕФЕРАТ

Выпускная квалификационная работа по теме “Backend API для приложения SensoRehab компании Sensomed” содержит 43 страницы текстового документа, 1 приложение, 10 использованных источников, 21 рисунок.

ПЕРЧАТКА, СЕНСОМЕД, SENSOREHAB, ДАННЫЕ, API.

Цель работы: разработка backend API для регистрации, входа, работы клиник и хранения данных приложения “SensoRehab”.

В результате выполнения ВКР было разработано приложение «SensoRehab-Backend».

В выпускную квалификационную работу входит введение, 3 главы и заключение.

Во введении ставится цель и выполняется ее декомпозиция на задачи.

В первой главе рассматриваются аналоги, формулируются основные требования к разрабатываемому приложению и выбираются инструменты разработки.

Во второй главе разрабатываются: архитектура системы, структура базы данных и структуры данных.

В третьей главе описана реализация основных элементов приложения.

В заключении подводятся итоги по выполненной работе.

СОДЕРЖАНИЕ

Введение 3
1 Анализ технического задания 5
1.1 Анализ существующих решений 6
1.2 Спецификация требований на приложение 6
1.3 Анализ и выбор инструментов разработки 9
1.4 Выводы по разделу 10
2 Этапы проектирования 11
2.1 Прецеденты 11
2.2 База данных 21
2.3 Алгоритмы приложения 26
2.3.1 Алгоритм работы приложения 26
2.3.2 Алгоритм завершения работы приложения 28
2.3.3 Алгоритм очистки просроченных кодов подтверждения 29
2.3.4 Алгоритм обновления данных пациента 30
2.4 Архитектура приложения 31
2.5 Выводы по главе 31
3 Программная реализация приложения 32
3.1 Написание маршрутов и контроллеров 32
3.2 Конечные точки доступа 33
3.3 Окружение и инициализация приложения 37
3.4 Проверка доступа к объекту 38
3.5 Завершение приложений 38
Заключение 39
Список использованных источников 40
Приложение А. Акт о внедрении 41

ВВЕДЕНИЕ

Инсульт является одним из наиболее серьезных заболеваний, встречающихся в Российской Федерации, возникает более чем у 500 тысяч человек ежегодно. Примерное количество подобных случаев в Москве – 40 тысяч, в Санкт-Петербурге – 25 тысяч, а в Красноярске около 12 тысяч в год. Инсульт не только приводит к значительной смертности, но и оставляет выживших пациентов, среди которых около 20% пациенты с тяжелой инвалидностью и потребностью постоянного ухода, ограниченно трудоспособны 56% и только 8% возвращаются к своей прежней работе [1][2].

Для увеличения мелкой моторики, уменьшения спастики и реабилитации после инсульта, необходимы устройства. Это может быть эспандер, пальчиковые тренажеры или другие различные тренажеры.

Продукт «SensoRehab» от компании «Сенсомед» предлагает решение для данной проблемы. SensoRehab объединяет в себе всю вариативность упражнений (сведение и разведение пальцев, смыкание и размыкание указательного и большого пальца, раскрытие и удержание пальцев кисти, ладонное и тыльное сгибание кисти, супинация и пронация предплечья), графиков и игр.

Такие устройства часто приобретают медицинские клиники, реабилитационные центры и отдельные физические лица в качестве покупателей.

Так как приложение позволяет регистрироваться в приложении пользователям, играть в игры, сохранять результаты измерений, у клиник есть возможность добавлять пациентов, просматривать и редактировать их личные данные.

Для того, чтобы централизованно хранить и обрабатывать данные, на помощь приходит Backend API, которое будет принимать, обрабатывать, хранить и отправлять данные клиентской части приложения.

Целью работы является разработка серверного веб приложения. Разрабатываемое приложение представляет собой WEB Backend API, которое будет принимать, обрабатывать и хранить пользовательские данные. Приложение должно давать непрерывный доступ пользователю к его данным для их синхронизации между различными устройствами. Для достижения цели в работе решаются следующие задачи:

– проанализировать техническое задание на разработку приложения «SensoRehab-Backend»;

– выполнить проектирование ПО;

– выполнить разработку ПО.

1 Анализ технического задания

Backend API – это набор инструкций и правил, которые позволяют различным программным приложениям взаимодействовать друг с другом. API определяет способ обмена данными и командами между различными программами, в основном между пользовательской и серверной частью приложения.

Ключевые аспекты роли Backend API:

– предоставление данных: Backend API отвечает за доступ к данным, хранящимся на сервере или в базе данных. Он может предоставлять клиентам приложения доступ к различным типам данных, таким как текст, изображения, аудио- и видеофайлы, а также структурированные данные в формате JSON или XML;

– бизнес-логика и обработка запросов: Backend API выполняет бизнес-логику приложения, обрабатывает запросы от клиентов, осуществляет валидацию данных, выполнение различных операций и вычислений, а также принимает решения на основе внутренней логики и правил приложения;

– аутентификация и авторизация: Backend API обеспечивает механизмы аутентификации и авторизации пользователей. Он может проверять учётные данные, выделять и проверять токены доступа, контролировать права доступа пользователей к определенным ресурсам и функциональности приложения;

– интеграция с внешними сервисами: Backend API может взаимодействовать с другими внешними сервисами, API и системами, чтобы получать или отправлять данные, вызывать функции или получать информацию. Это может включать интеграцию с социальными сетями, платежными системами, почтовыми службами и другими сторонними сервисами.

1.1 Анализ существующих решений

Среди прямых конкурентов разрабатываемого программного обеспечения SensoRehab можно выделить Anika [3], Rapael Smart Glove [4] и Hand Tutor [5]. Все вышеперечисленные продукты — тренажеры с биологической обратной связью, помогающие восстановить мелкую моторику и координацию движений.

Как и SensoRehab, конкурентные продукты используются в реабилитации пациентов с повреждениями головного и спинного мозга, для восстановления моторики рук после операций и травм, при детском церебральном параличе (ДЦП) и болезни Паркинсона, а также при реабилитации после инсульта.

Продукт SensoRehab использует три канала биологической обратной связи (аудиальный, визуальный и тактильный), в то время как Rapael Smart Glove и Anika используют только два канала обратной связи (аудиальный и визуальный).

У людей с нарушенной мелкой моторикой, с точки зрения работы с приложением, возникнет меньше проблем при использовании SensoRehab, так как там значительно лучше продуман дизайн подключения, конфигурации и калибровки перчатки, чем у остальных конкурентов.

Также у людей с ограниченными физическими возможностями могут возникнуть проблемы с надеванием устройств конкурентов, так как они недостаточно эргономичны, в отличии от продукта SensoRehab.

1.2 Спецификация требований на приложение

Необходимо реализовать API-запросы для взаимодействия с пользователем и правильной работы приложения (Рисунок 1).

Требования к возможностям взаимодействия с API:

– регистрация новой клиники;

– регистрация нового пациента;

– подтверждение номера телефона или электронной почты пользователя;

– вход в личный кабинет клиники;

– вход в личный кабинет пациента;

– сброс и восстановление пароля для личных кабинетов;

– получение списка пациентов клиники;

– изменение данных пациента клиники;

– создание записей о новых пациентах клиники;

– архивирование пациентов клиники;

– создание результатов измерений пациентов;

– архивирование результатов измерений пациентов;

– создание результатов игр пациентов;

– архивирование результатов игр пациентов.

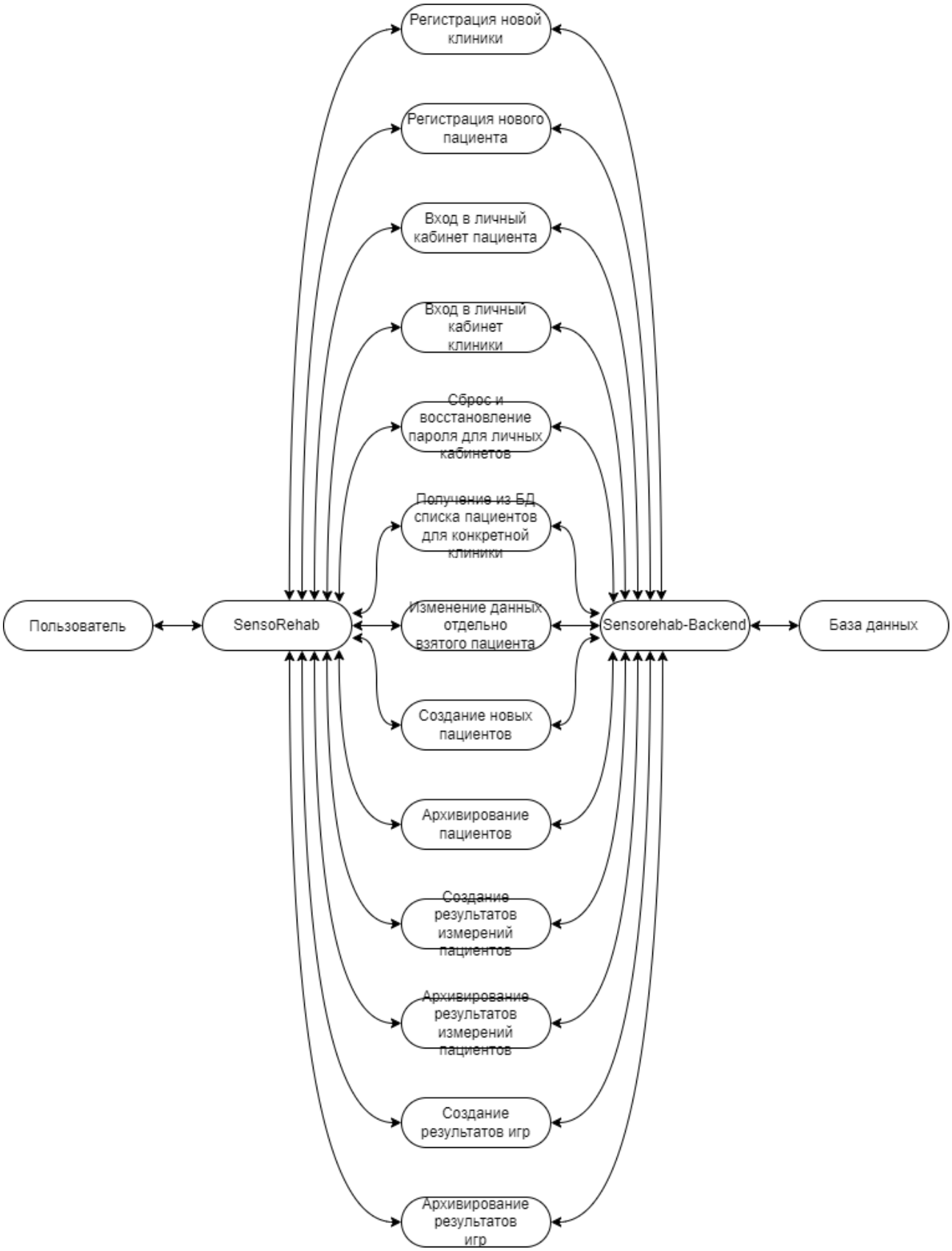

Рисунок 1 – Диаграмма вызовов API представляет взаимодействие пользователя с приложением «SensoRehab»

При разделении задачи разработки API «SensoRehab-Backend» на отдельные этапы были определены следующие подзадачи:

– создать структуру данных и архитектуру базы данных;

– создать миграцию для базы данных;

– реализовать связь между приложением и базой данных;

– создать функционал обращения к API сторонних сервисов для отправки сообщений на мобильный и почту пользователей для подтверждения аккаунтов и сброса пароля;

– создать асинхронную задачу, удаляющую просроченные коды для подтверждения регистрации или для сброса пароля;

– развернуть приложение внутри Docker контейнера [6];

– ограничить количество запросов, обрабатываемых одним экземпляром приложения;

– реализовать отправку сообщений об ошибках из приложения на сторонний сервис Sentry [7].

1.3 Анализ и выбор инструментов разработки

Для разработки приложения был выбран язык Python. Python имеет обширную стандартную библиотеку и множество сторонних библиотек, которые облегчают разработку, а также имеет большое и активное сообщество разработчиков.

Django, Flask и FastAPI [8] являются популярными фреймворками для разработки веб-приложений и API на языке Python. При выборе фреймворка основной упор ставился на высокую производительность и скорость разработки, чем отличается FastAPI от остальных фреймворков. Помимо этого, в FastAPI отсутствует функционал, который не нужен нашему приложению (публикация статичных страниц, панель администрирования и др.).

В качестве базы данных используется был выбран PostgreSQL [9], так как она является мощной реляционной базой данных с открытым исходным кодом. Также PostgreSQL имеет активное сообщество разработчиков, которые постоянно работают над улучшением базы данных и созданием новых расширений.

1.4 Выводы по разделу

Были рассмотрены подходящие для разрабатываемого приложения аналоги и проведен анализ предметной области, на основе которого были выявлены их сильные и слабые стороны, что позволило выделить главные требования к разрабатываемому приложению. Были проанализированы и выбраны инструменты разработки, оптимально подходящие для создания приложения.

2 Этапы проектирования

2.1 Прецеденты

Прецедент «Регистрация новой клиники» (Рисунок 2). Необходимо дать пользователю возможность зарегистрировать новую клинику. Пользователь должен открыть страницу регистрации клиники в приложении, ввести название клиники, почту и пароль. В случае успеха выполняется запрос к серверу, в результате возвращаются идентификатор новой клиники, пользователю отправляется код подтверждения на почту, открывается окно ввода кода. В ином случае есть два варианта. Сервер вернул необработанную ошибку, тогда отображается всплывающее окно с описанием ошибки. Пользователь ввёл некорректный почтовый адрес или пароль, тогда выводится сообщение об ошибке (Рисунок 3).

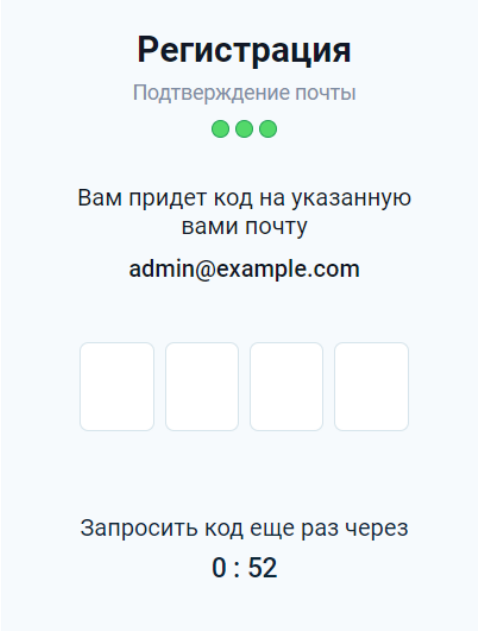

Рисунок 2 – Ответ с полем для ввода кода

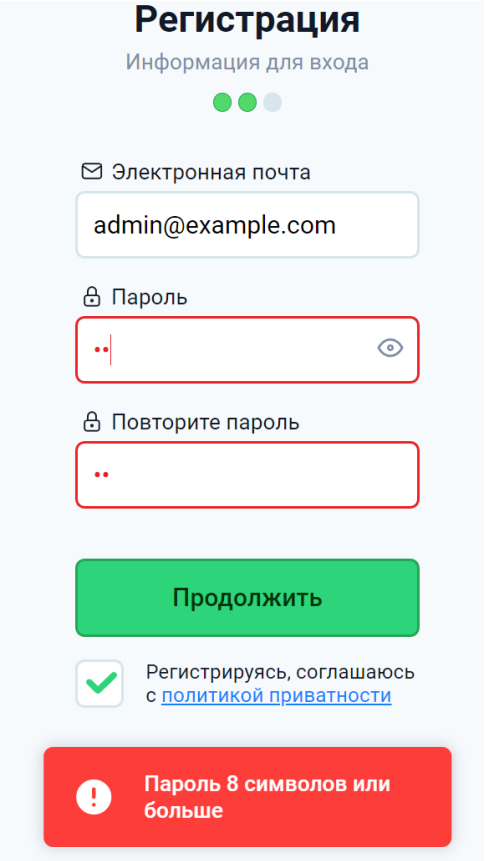

Рисунок 3 – Ответ с сообщением об ошибке

Прецедент «Регистрация нового пациента». Необходимо дать пользователю возможность зарегистрировать нового пациента. Пользователь должен открыть страницу регистрации пациента в приложении, ввести название клиники, почту и пароль. В случае успеха выполняется запрос к серверу, в результате возвращаются идентификатор новой клиники, пользователю отправляется код подтверждения на почту, открывается окно ввода кода (Рисунок 4). В ином случае есть два варианта. Сервер вернул необработанную ошибку, тогда отображается всплывающее окно с описанием ошибки. Пользователь ввёл некорректный номер телефона или пароль, тогда выводится сообщение об ошибке (Рисунок 5).

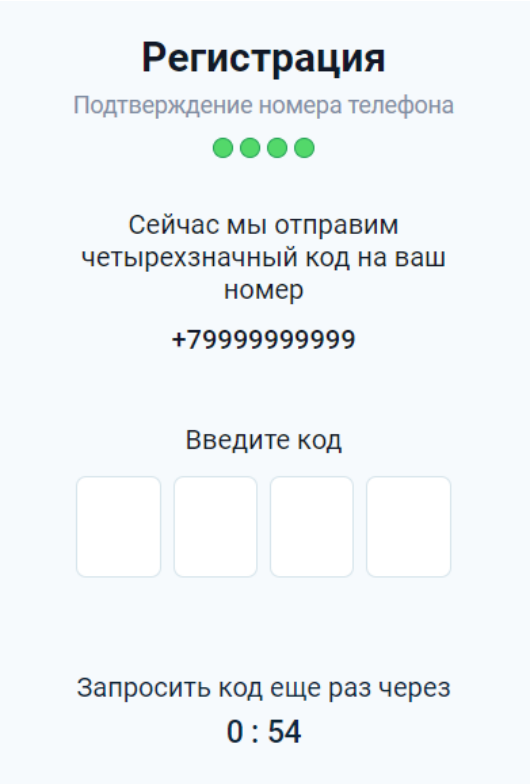

Рисунок 4 – Ответ с полем ввода для кода

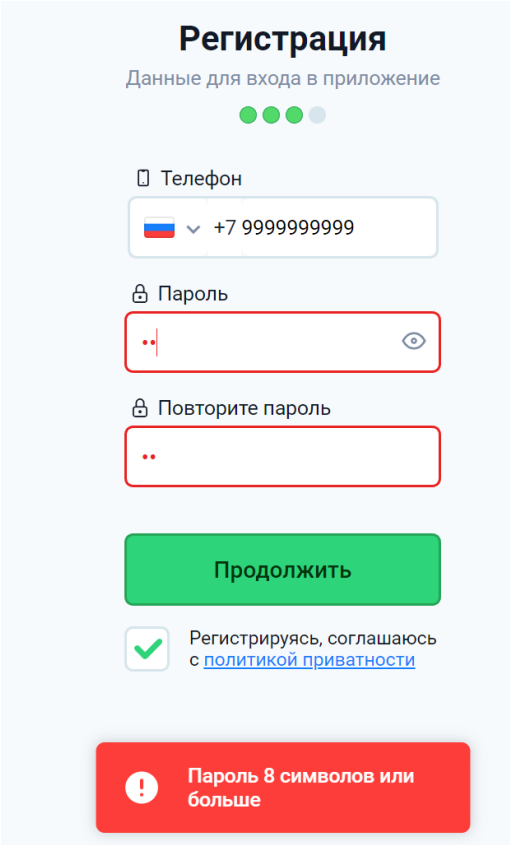

Рисунок 5 – Ответ с сообщением об ошибке

Прецедент «Подтверждение номера телефона или электронной почты пользователя». Необходимо подтвердить номер телефона пациента или электронную почту клиники. Пользователь должен ввести код подтверждения. В случае успеха выполняется запрос к серверу, в результате возвращается идентификатор подтвержденного пользователя, открывается окно входа в приложение. В ином случае есть два варианта. Сервер вернул необработанную ошибку, тогда отображается всплывающее окно с описанием ошибки. Пользователь ввёл неверный код, тогда отображается всплывающее окно с сообщением о неправильном коде, пользователь остается на странице ввода кода (Рисунок 6).

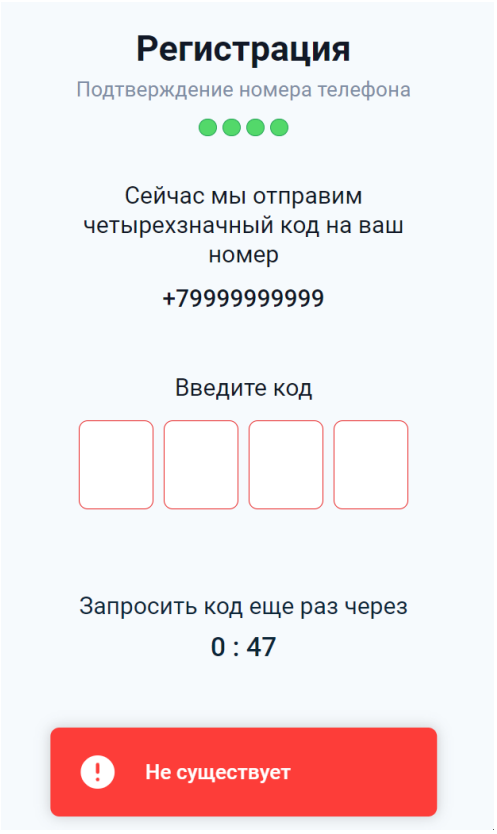

Рисунок 6 – Ответ с сообщением об ошибке

Прецедент «Вход в личный кабинет клиники». Необходимо дать пользователю войти в личный кабинет клиники. Пользователь должен открыть страницу входа в личный кабинет клиники. В случае успеха выполняется запрос к серверу, в результате возвращаются объект клиники и токен

авторизации, открывается окно личного кабинета клиники со списком её пациентов. В ином случае есть два варианта. Сервер вернул необработанную ошибку, тогда отображается всплывающее окно с описанием ошибки. Пользователь ввёл неправильную электронную почту или пароль, тогда отображается всплывающее окно с сообщением о том, что пользователь ввёл неправильную электронную почту или пароль (Рисунок 7).

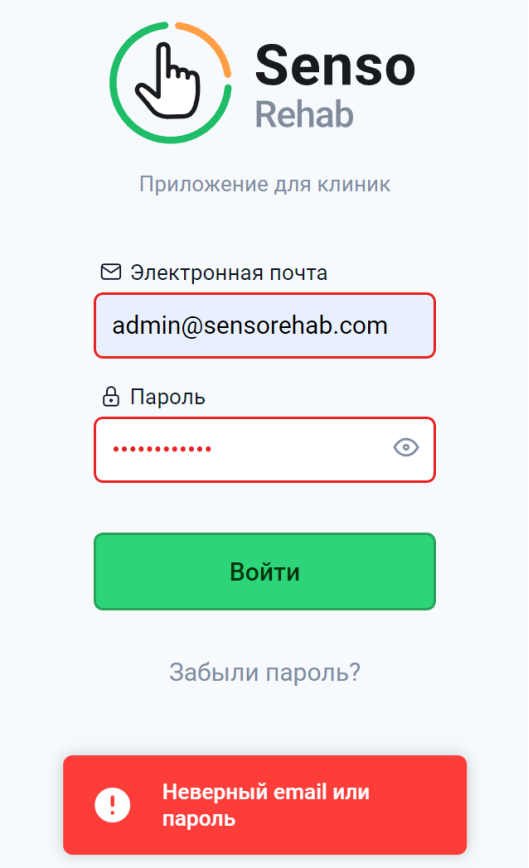

Рисунок 7 – Ответ с сообщением об ошибке

Прецедент «Вход в личный кабинет пациента». Необходимо дать пользователю войти в личный кабинет пациента. Пользователь должен открыть страницу входа в личный кабинет пациента. В случае успеха выполняется запрос к серверу, в результате возвращаются объект пациента и токен авторизации, открывается окно личного кабинета пациента. В ином случае есть два варианта. Сервер вернул необработанную ошибку, тогда отображается всплывающее окно с описанием ошибки. Пользователь ввёл неправильный

номер телефона или пароль, тогда отображается всплывающее окно с сообщением о том, что пользователь ввёл неправильный номер телефона или пароль (Рисунок 8).

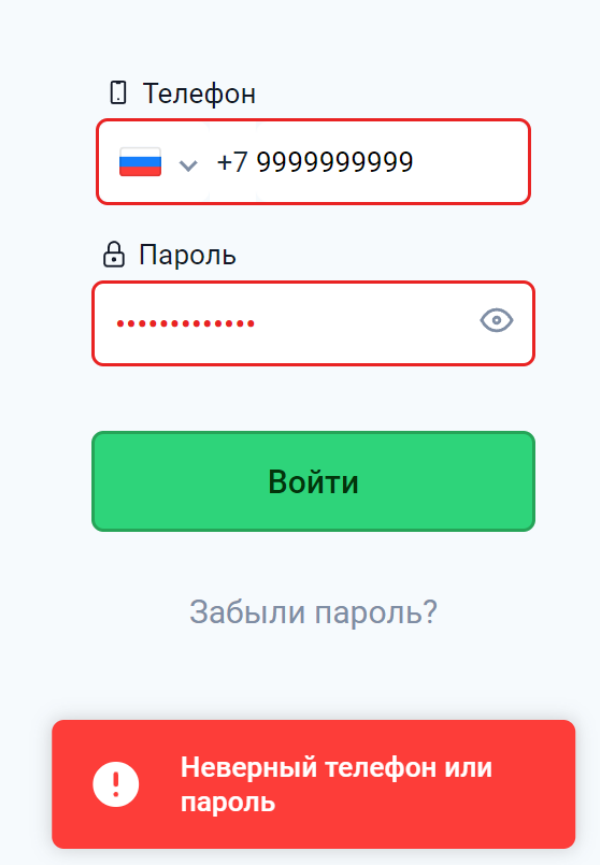

Рисунок 8 – Ответ с сообщением об ошибке

Прецедент «Сброс и восстановление пароля для личных кабинетов». Необходимо дать пользователю сбросить пароль. Пользователь должен открыть страницу сброса и восстановления пароля и ввести свой номер телефона или электронную почту. В случае успеха выполняется запрос к серверу с почтой или номером телефона, пользователю отправляется код для подтверждения, открывается страница для ввода кода подтверждения. В ином случае есть два варианта. Сервер вернул необработанную ошибку, тогда отображается всплывающее окно с описанием ошибки. Пользователь ввёл несуществующий номер телефона или электронную почту, тогда отображается всплывающее

окно с сообщением, что данный номер телефона или электронная почта не существуют (Рисунок 9).

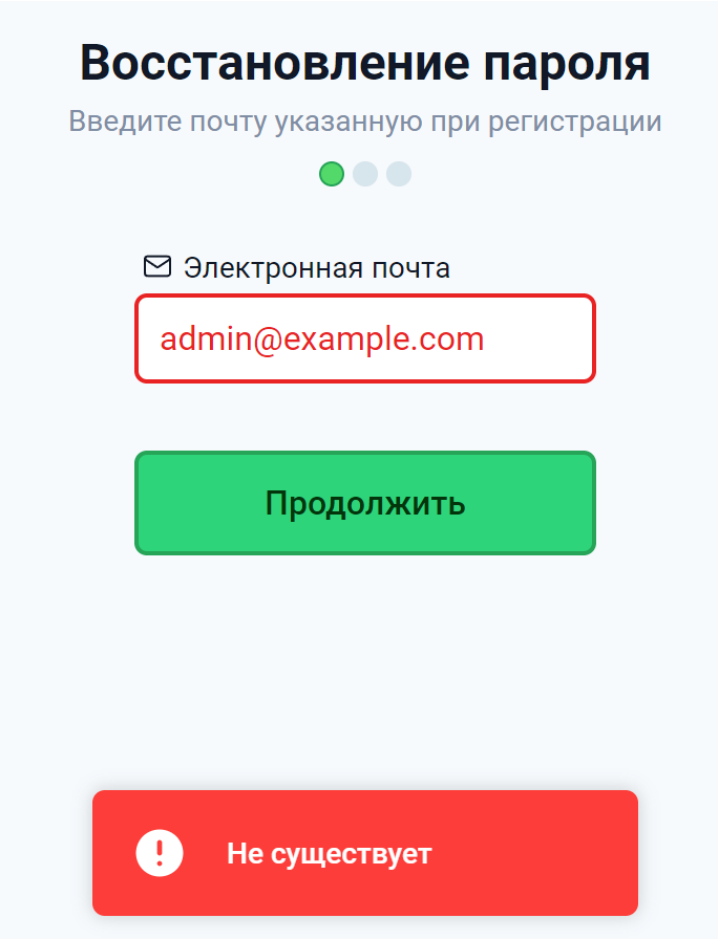

Рисунок 9 – Ответ с сообщением об ошибке

Прецедент «Получение списка пациентов клиники» (Рисунок 10). Необходимо дать пользователю клиники получить данные о её пациентах. Пользователь должен войти в личный кабинет клиники. В случае успеха отправляется запрос на сервер с токеном авторизации клиники и начальным id пациента, пользователю возвращается список пациентов, который начинается с пациента с начальным id, максимальный размер которого 1000 пациентов. У пользователя отображается список его пациентов. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

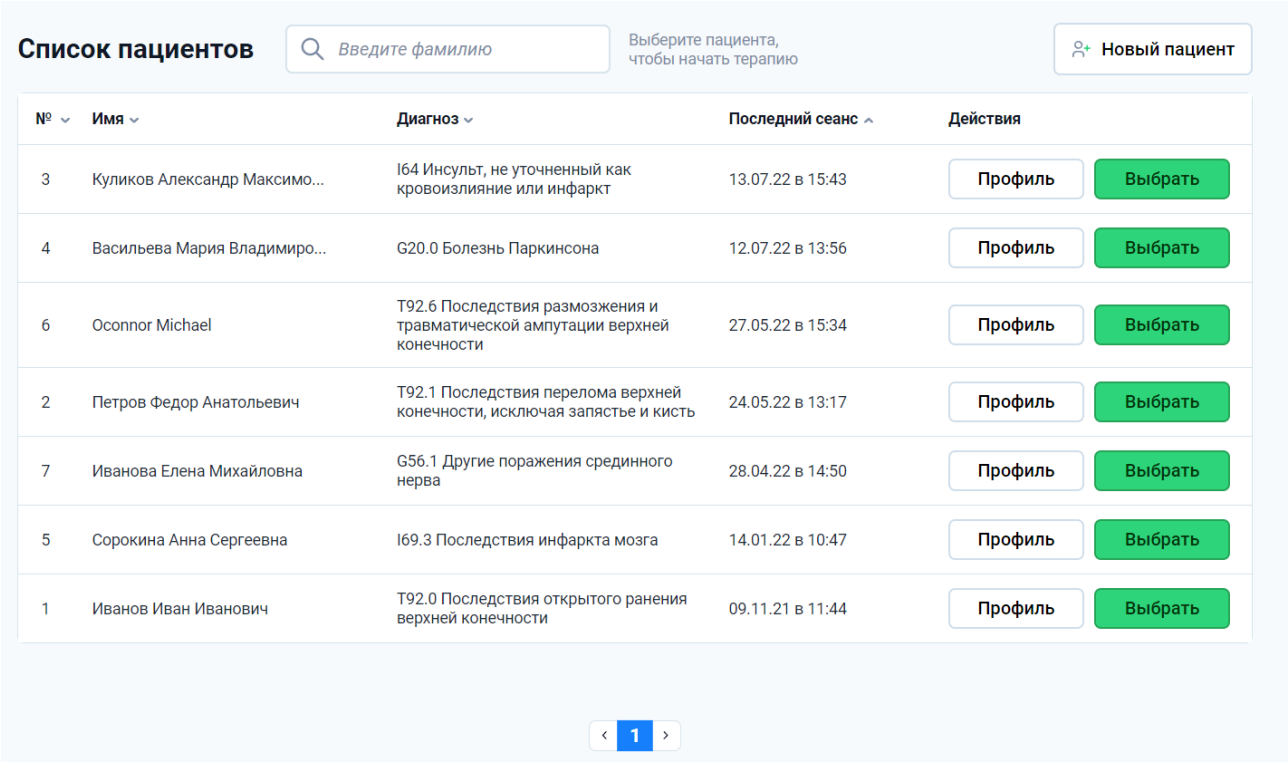

Рисунок 10 – Ответ со списком пациентов

Прецедент «Изменение данных пациента клиники» (Рисунок 11). Необходимо дать клинике изменять личные данные её пациента. Пользователь должен открыть окно изменения личных данных пациента. В случае успеха выполняется запрос к серверу, в результате возвращаются id изменённого пациента. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

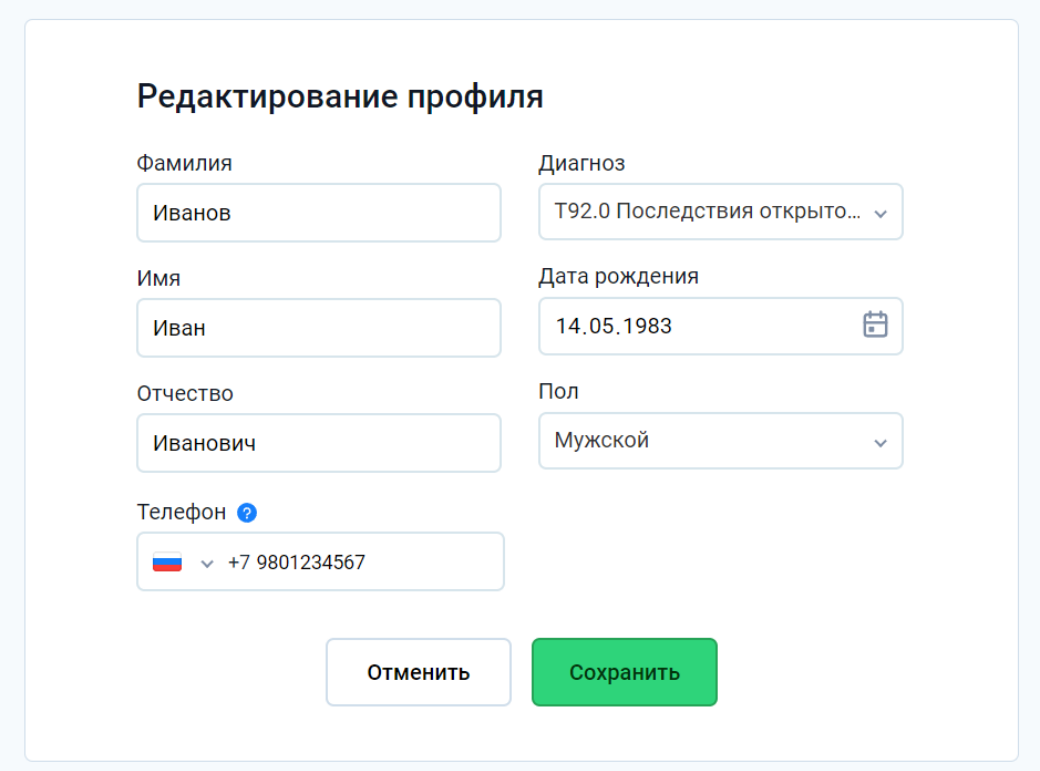

Рисунок 11 – Окно изменения данных пациента

Прецедент «Создание новых пациентов клиники» (Рисунок 12). Необходимо дать пользователю клиники создавать нового пациента. Пользователь должен открыть окно создания нового пациента. В случае успеха отправляется запрос на сервер, возвращаются id созданных пациентов. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

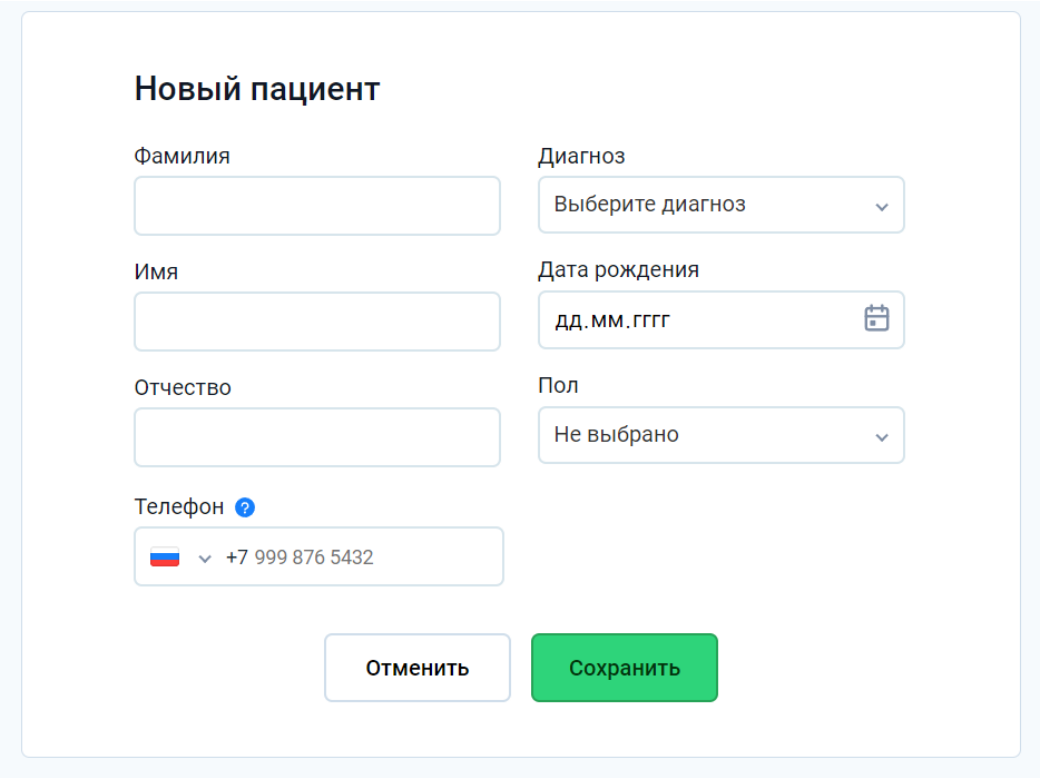

Рисунок 12 – Окно создания нового пациента

Прецедент «Архивирование пациентов клиники». Необходимо дать пользователю клиники архивировать личные данные пациента. Пользователь должен открыть окно профиля пациента и нажать на кнопку архивации. В случае успеха отправляется запрос на сервер, на сервере результаты измерений и игр пациентов архивируются каскадом, возвращается id заархивированных пациентов. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

Прецедент «Создание результатов измерений пациентов». Необходимо дать пользователю возможность создать результаты измерения амплитуды движения руки. Пользователь должен запустить измерение амплитуды движения руки пациента перед игрой. В случае успеха выполняется запрос к серверу, в результате возвращаются id созданных результатов измерений пациентов. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

Прецедент «Архивирование результатов измерений пациентов». Необходимо дать пользователю архивировать результаты измерений пациента.

Пользователь должен открыть окно со списком измерений пациента и нажать на кнопку архивации. В случае успеха выполняется запрос к серверу, в результате возвращаются id изменённых результатов измерений. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

Прецедент «Создание результатов игр пациентов». Необходимо дать пользователю создавать результат игры. Пользователь должен сыграть в игру. В случае успеха выполняется запрос к серверу, в результате возвращаются id созданных результатов игр. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

Прецедент «Архивирование результатов игр». Необходимо дать пользователю архивировать результаты игр пациента. Пользователь должен открыть окно со списком результатов игр пациента и нажать на кнопку архивации. В случае успеха выполняется запрос к серверу, в результате возвращаются id изменённых результатов измерений. В ином случае сервер вернёт необработанную ошибку, тогда отобразится всплывающее окно с описанием ошибки.

2.2 База данных

При проектировании архитектуры баз данных, была выбрана третья нормальная форма базы. Она помогает устранить избыточность данных и обеспечить эффективность при выполнении запросов.

При проектировании учитывалось, что у разных клиник могут быть пациенты с одинаковыми номерами телефонов, что пациент с подтвержденным номером телефона может быть только один. Также было учтено, что результат измерения может быть без результата игры (например, если игра не была окончена), а результат игры не может быть без результата измерений, потому что в момент измерения происходит калибровка устройства, без которой игровой процесс невозможен. Помимо этого, также было замечено, что у

одного измерения, могут быть разные диапазоны движений на разных датчиках, поэтому отношение результата измерения к диапазону движений, это отношение один ко многим.

Таблицы archive_status, sex, confirmation_code_type, exercise, sensor, game, difficulty_level, difficulty_type, diagnosis:

– id – идентификатор константы;

– name – имя константы.

Данные таблицы являются таблицами констант. При помощи данных таблиц упрощается работа, другие таблицы используют уникальный идентификатор константы, вместо его имени.

Таблица phone:

– id – идентификатор номера телефона;

– number – номер телефона;

– is_confirmed – булевое значение, является ли телефон подтвержденным.

Данная таблица описывает номера телефонов, которые указаны в профилях пациентов клиники и частных пациентов.

Таблица email:

– id – идентификатор электронной почты;

– address – адрес электронной почты;

– is_confirmed – булевое значение, является ли адрес подтвержденным.

Данная таблица описывает адреса электронных почт, которые указаны в профилях клиник.

Таблица clinic:

– id – идентификатор клиники;

– name – название клиники;

– password – хэш пароля;

– email_id – идентификатор почтового адреса.

Данная таблица описывает клинику, хранит хэш пароля, который сверяется при каждой авторизации, с помощью таблицы frontend получает данные клиники при входе в неё.

Таблица certificate:

– id – идентификатор сертификата;

– patient_id – идентификатор пациента;

– clinic_id – идентификатор клиники.

Данная таблица описывает сертификаты, отношения с клиникой и пациентом, с которым происходит процедура восстановления пароля.

Таблица confirmation_code:

– id – идентификатор кода подтверждения;

– value – значение кода подтверждения;

– call_id – идентификатор запроса стороннего сервиса;

– email_id – идентификатор почтового адреса;

– phone_id – идентификатор номера телефона;

– confirmation_code_type_id – идентификатор-константа тип кода подтверждения, хранит данные о том, для чего используется этот код, восстановление пароля или регистрация.

Данная таблица описывает коды подтверждения, отношения с номером телефона и почтовым адресом, с которым происходит процедура регистрации или восстановления пароля.

Таблица patient:

– id – идентификатор пациента;

– first_name – имя пациента;

– middle_name – отчество пациента;

– last_name – фамилия пациента;

– password – хэш пароля;

– birth_date – дата рождения пациента;

– last_session – дата и время, когда в последний раз занимался пациент;

– phone_id – идентификатор номера телефона;

– clinic_id – идентификатор клиники, к которой привязан пациент;

– sex_id – идентификатор-константа пола;

– diagnosis_id – идентификатор-константа диагноз;

– archive_status_id – идентификатор-константа статуса архивации;

– comment – комментарий о пользователе.

Данная таблица описывает пациента, его отношения с клиникой, хранит хэш пароля, который сверяется при каждой авторизации, с помощью таблицы frontend получает данные частного пациента при входе в него, а данные пациентов клиники получаются при помощи асинхронных задач.

Таблица measurement:

– id – идентификатор измерения;

– patient_id – идентификатор пациента;

– exercise_id – идентификатор-константа упражнения, которым занимался пациент;

– clinic_id – идентификатор клиники;

– archive_status_id – идентификатор-константа статуса архивации.

Данная таблица описывает измерение, его отношения с клиникой и пациентом, с помощью таблицы frontend получает данные измерения используя асинхронные задачи.

Таблица motion_range:

– id – идентификатор измерения;

– start – начало диапазона движений;

– end – конец диапазона движений;

– measurement_id – идентификатор измерения;

– sensor_id – идентификатор-константа сенсора, с которого был взят диапазон движений пациента.

Данная таблица описывает диапазон движения измерения, его отношения с измерением.

Таблица game_result:

– id – идентификатор результата игры;

– duration – длительность игры;

– score – количество очков, набранных во время игры;

– best_speed – максимальная скорость движения во время игры;

– patient_id – идентификатор пациента;

– game_id – идентификатор-константа, игра, в которую играл пациент;

– mean_speed – средняя скорость движений во время игры;

– difficulty_level_id – идентификатор-константа уровня сложности игры;

– difficulty_type_id – идентификатор-константа типа сложности игры;

– clinic_id – идентификатор клиники;

– archive_status_id – идентификатор-константа статуса архивации;

– exercise_id – идентификатор-константа упражнения, которым занимался пациент;

– measurement_id – идентификатор измерения.

Данная таблица описывает результат игры, его отношения с клиникой, пациентом и результатом измерения, с помощью таблицы frontend получает данные результата игры используя асинхронные задачи.

На основе имеющихся таблиц, а также первичных и внешних ключей была создана ER-диаграмма, на которой показаны связи между моделями базы данных (Рисунок 13). С диаграммы были удалены вспомогательные таблицы, для более лучшего восприятия.

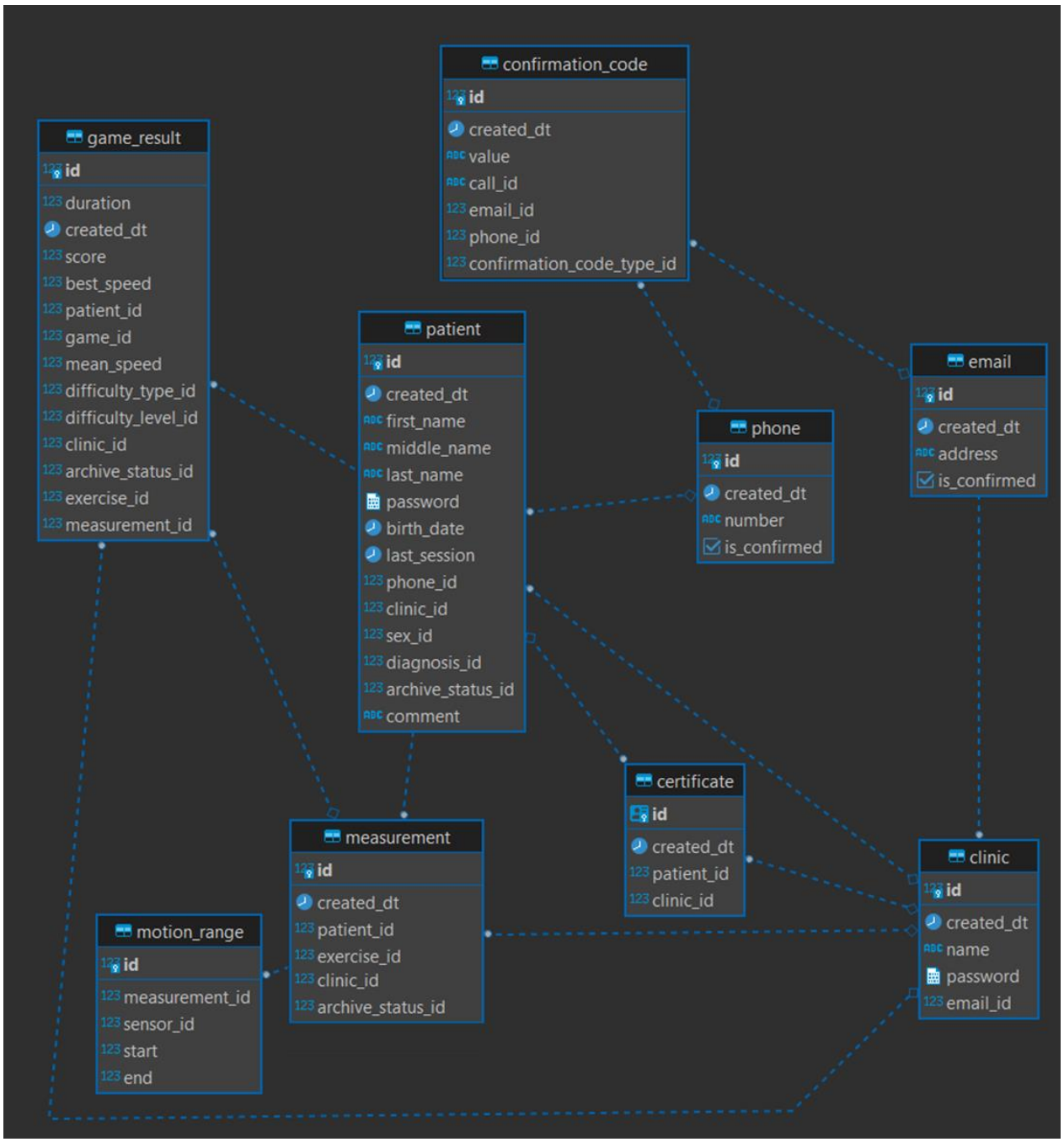

Рисунок 13 – ER Диаграмма приложения SensoRehab-Backend

2.3 Алгоритмы приложения

2.3.1 Алгоритм работы приложения

Приложение начинает работу с инициализации после запуска. Во время инициализации, приложение извлекает необходимые данные из переменных окружения контейнера, которые понадобятся для его работы. После успешной

инициализации, приложение переходит к обработке системных сигналов в параллельном потоке, чтобы обеспечить правильное завершение работы приложения в случае возникновения ошибок или неожиданной ситуации.

В основном потоке приложение ожидает запросы пользователей. При получении запроса, сначала происходит проверка доступа пользователя, чтобы убедиться, что он имеет необходимые права для выполнения запроса. Затем происходит обработка запроса, в ходе которой приложение выполняет необходимые действия в соответствии с полученными данными.

После обработки запроса приложение генерирует ответ, который отправляется пользователю. После отправки ответа приложение готово к приему и обработке следующего запроса, продолжая работу в цикле ожидания новых запросов и обработки в соответствии с описанным алгоритмом (Рисунок 14).

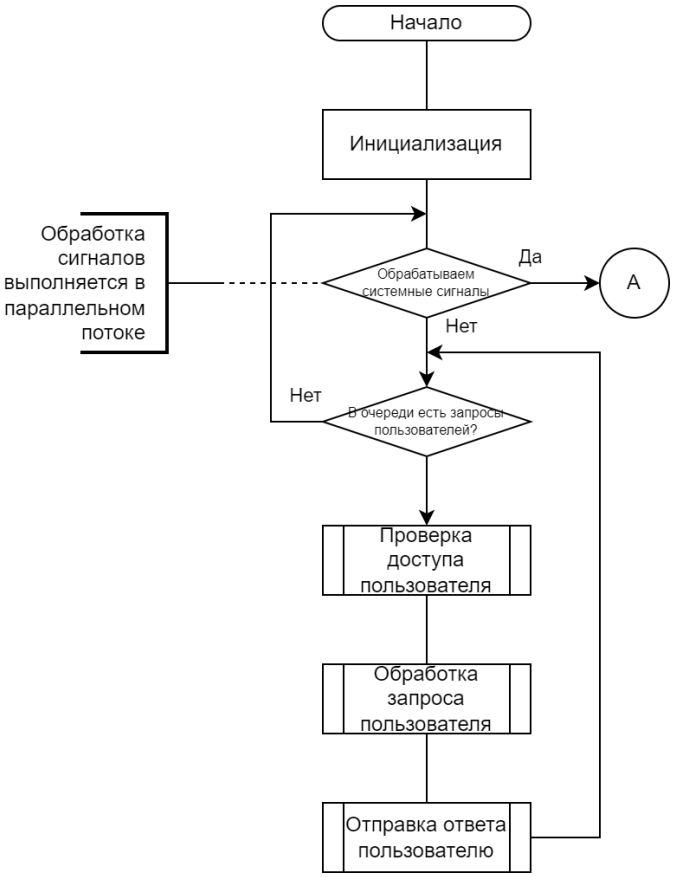

Рисунок 14 – Алгоритм работы приложения

2.3.2 Алгоритм завершения работы приложения

Приложение выполняет необходимые действия для корректного завершения работы, такие как очистка ресурсов и закрытие сетевых соединений (Рисунок 15). После завершения этих операций приложение закрывается и освобождает занимаемые ресурсы, чтобы предотвратить утечки памяти или другие проблемы. В итоге, алгоритм завершения работы приложения гарантирует, что оно корректно завершает операции и освобождает ресурсы перед закрытием.

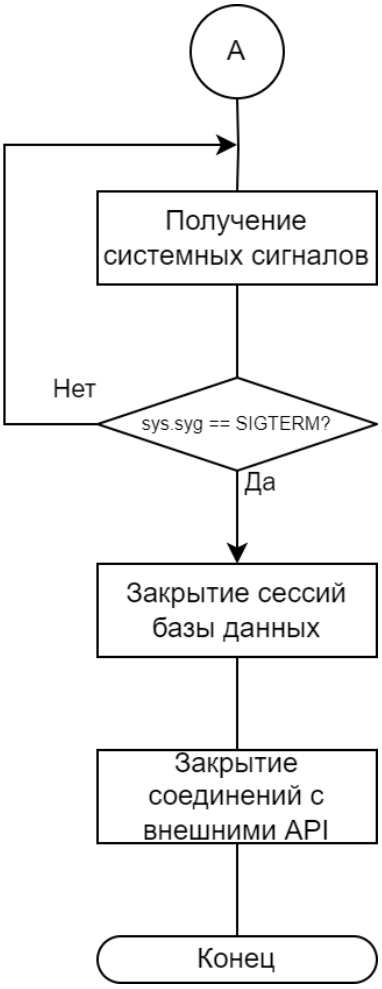

Рисунок 15 – Алгоритм завершения работы приложения

2.3.3 Алгоритм очистки просроченных кодов подтверждения

Записи в таблице кодов подтверждения могут быть просрочены по нескольким причинам:

– пользователь не закончил процесс регистрации;

– пользователю не пришел код подтверждения, и он запросил новый;

– пользователь вспомнил пароль и прекратил процедуру восстановления пароля досрочно.

Чтобы не хранить все просроченные коды подтверждения, был создан алгоритм очистки кодов (Рисунок 16).

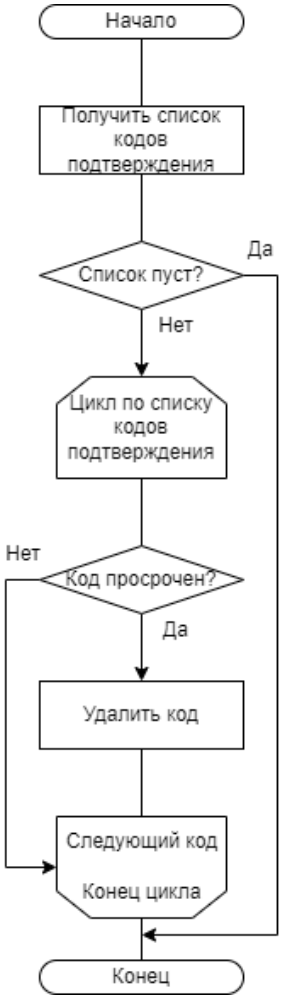

Рисунок 16 – Алгоритм очистки просроченных кодов подтверждения

2.3.4 Алгоритм обновления данных пациента

Данные пациента могут обновляться с нескольких устройств одновременно. Также frontend приложение может работать без доступа к интернету. Поэтому была введена система версий объекта пациента и алгоритм обновления данных (Рисунок 17).

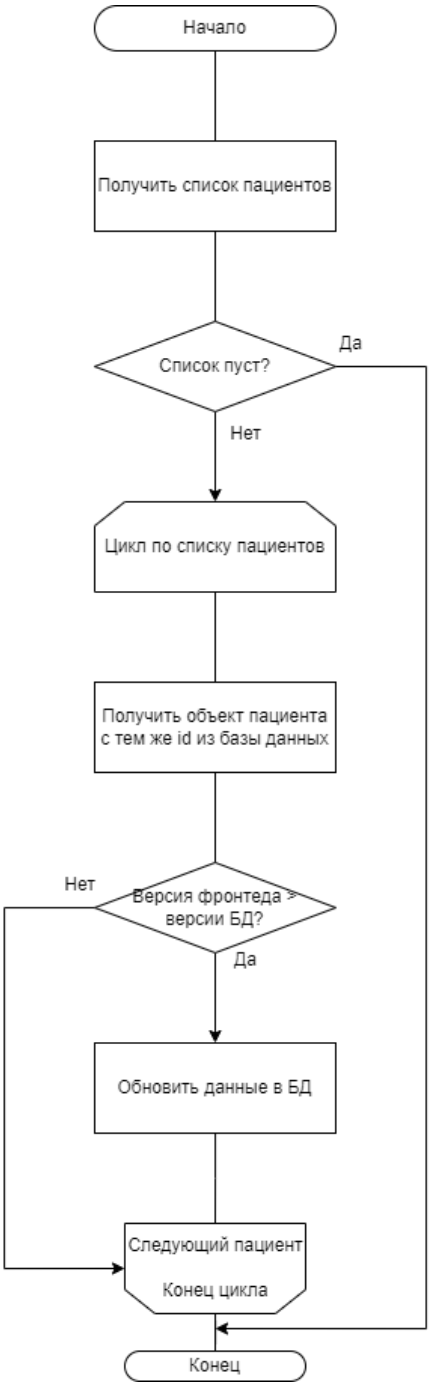

Рисунок 17 – Алгоритм обновления данных пациента

2.4 Архитектура приложения

Существует несколько различных видов архитектуры Backend API приложений, каждая из которых имеет свои преимущества и недостатки.

Два из наиболее распространенных подходов – монолитная архитектура и микросервисная архитектура.

Монолитная архитектура – это классический подход, при котором все компоненты приложения находятся в одном цельном блоке. Он включает в себя базу данных, бизнес-логику и пользовательский интерфейс внутри одной монолитной кодовой базы. В этой модели все запросы от клиентов обрабатываются внутри одного приложения.

С другой стороны, микросервисная архитектура разбивает приложение на небольшие, изолированные сервисы, каждый из которых отвечает за выполнение определенной функциональности. Каждый микросервис может быть разработан, развернут и масштабирован независимо от других. Каждый сервис может иметь свою собственную базу данных и связь с другими сервисами происходит посредством API вызовов.

Так как необходима простота и быстрота разработки, монолитная архитектура – отличный выбор для разработки приложения. В приложении нет необходимости, разбивать его на отдельные сервисы, так как все данные зависят друг от друга и нет отдельных задач, которые можно будет отделить в другой сервис.

2.5 Выводы по главе

Согласно требованиям технического задания, была предложена структура данных и архитектура базы данных. На представленных выше рисунках показаны алгоритмы, которые отражают наиболее сложные взаимосвязи в системе и более подробно проработано взаимодействие между объектами. Также была выбрана архитектура приложения.

3 Программная реализация приложения

3.1 Написание маршрутов и контроллеров

Как упоминалось ранее, основной целью является создание API, которое позволяет пользователям создавать, изменять и получать объекты данных, связанные с их профилем.

Определение маршрутов и контроллеров включает в себя разработку URL-адресов (маршрутов), которые клиенты API могут использовать для выполнения операций с ресурсами, и написание соответствующих контроллеров для обработки запросов и взаимодействия с данными.

При разработке маршрутов и контроллеров используется язык программирования Python 3.10, фреймворки FastAPI и SQLAlchemy. SQLAlchemy помогает при создании миграций, с работой с базой данных и выполняет роль ORM, а FastAPI выполняет роль маршрутизатора, который определяет и создает конечные точки.

В файле routes.py прописываются все правила маршрутизации URL-адресов (Рисунок 18).

```
async def init(app: fastapi.FastAPI):
    _ROUTER_CONFIGS = [
        {"router": views.v1.auth.SignIn().router, "prefix": "/api/v1/auth/sign_in"},
        {"router": views.v1.auth.ClinicSignUp().router, "prefix": "/api/v1/auth/sign_up/clinic/credentials"},
        {"router": views.v1.auth.PatientSignUpCredentials().router, "prefix": "/api/v1/auth/sign_up/patient/credentials"},
        {"router": views.v1.auth.PatientSignUpAdditionalInfo().router, "prefix": "/api/v1/auth/sign_up/patient/additional"},
        {"router": views.v1.auth.ResetPassRequest().router, "prefix": "/api/v1/auth/reset-password/request"},
        {"router": views.v1.auth.ResetPasswordNew().router, "prefix": "/api/v1/auth/reset-password/new"},
        {"router": views.v1.auth.Confirm().router, "prefix": "/api/v1/auth/confirm"},
        {"router": views.v1.auth.ResetPasswordCertify().router, "prefix": "/api/v1/auth/reset-password/certify"},
        {"router": views.v1.Patients().router, "prefix": "/api/v1/patients"},
        {"router": views.v1.GameResults().router, "prefix": "/api/v1/game_results"},
        {"router": views.v1.Measurements().router, "prefix": "/api/v1/measurements"},
    ]

    for router_config in _ROUTER_CONFIGS:
        app.include_router(**router_config)
```

Рисунок 18 – Пример маршрутизатора

3.2 Конечные точки доступа

Конечные точки доступа API являются важным элементом взаимодействия между различными приложениями и системами. Они представляют собой точки входа, через которые клиентские приложения могут получать доступ к функциональности или данным, предоставляемым определенным веб-сервисом или сервером:

а) **/api/v1/auth/sign_in** – необходима для входа в личный кабинет, содержит в себе POST метод, который возвращает в теле ответа токен авторизации и личные данные пользователя;

б) **/api/v1/auth/sign_up/clinic/credentials** – необходима для регистрации аккаунта клиники, содержит в себе метод POST, который принимает на вход название клиники, адрес электронной почты и пароль. В теле ответа возвращает идентификатор зарегистрированной клиники;

в) **/api/v1/auth/sign_up/patient/credentials** – необходима для регистрации аккаунта пациента, содержит в себе метод POST, который принимает на вход номер телефона и пароль. В теле ответа возвращает идентификатор зарегистрированного пациента;

г) **/api/v1/auth/sign_up/patient/additional** – необходима для получения личных данных пациента на стадии регистрации, содержит в себе метод POST, который принимает на вход его ФИО, пол, диагноз и дату рождения. В теле ответа возвращает идентификатор зарегистрированного пациента;

д) **/api/v1/auth/confirm** – необходима для подтверждения номера телефона или электронной почты после регистрации, содержит в себе метод POST, который принимает на вход код, который пришел пользователю. В теле ответа возвращает идентификатор пользователя;

е) **/api/v1/auth/reset-password/request** – необходима для отправки запроса на сброс пароля, содержит в себе метод POST, который принимает на вход номер телефона или электронную почту пользователя. В теле ответа возвращает идентификатор пользователя, сделавшего запрос на сброс пароля;

ж) **/api/v1/auth/reset-password/certify** – необходима для ввода кода, пришедшего пользователю после отправки запроса на сброс пароля, содержит в себе метод POST, который принимает на вход код, который пришел пользователю. В теле ответа возвращает идентификатор пользователя и сертификат, для подтверждения личности пользователя при создании нового пароля;

з) **/api/v1/auth/reset-password/new** – необходима для ввода нового пароля после его сброса, содержит в себе метод POST, который принимает на вход сертификат пользователя и новый пароль. В теле ответа возвращает идентификатор пользователя;

и) **/api/v1/patients** – необходима для работы с объектами пациентов, содержит в себе несколько методов:

1) POST **/api/v1/patients** – необходима для создания новых пациентов клиники. Принимает на вход список данных новых пациентов, в которые входят: ФИО, пол, диагноз и дату рождения. В теле ответа возвращает список идентификаторов созданных пациентов;
2) GET **/api/v1/patients** – необходима для получения данных о пациентах клиники. Принимает на вход необязательный параметр start_with_id, который необходим для пагинации. В теле ответа возвращает список данных пациентов, в которые входят: идентификатор пациента, его ФИО, пол, диагноз и дату рождения;
3) PATCH **/api/v1/patients** – необходима для изменения данных о пациенте. Принимает на вход список данных пациентов, в которые входят: идентификатор пациента, необязательные параметры ФИО, пол, диагноз и дата рождения и статус архивации. В теле ответа возвращает список идентификаторов измененных пациентов;

к) **/api/v1/game_results** – необходима для работы с объектами результатов игр, содержит в себе несколько методов:

1) POST **/api/v1/game_results** – необходима для создания результатов игр. Принимает на вход список данных новых результатов игр, в

которые входят: длительность, дата создания, количество очков, лучшую скорость движения, средняя скорость, идентификаторы пациента, игры, типа сложности, уровня сложности, клиники, статуса архивации упражнения и измерения. В теле ответа возвращает список идентификаторов созданных результатов игр;

2) GET **/api/v1/game_results** – необходима для получения данных о результатах игр. Принимает на вход необязательный параметр start_with_id, который необходим для пагинации. В теле ответа возвращает список данных результатов игр, в которые входят: идентификатор результата игры, длительность, дата создания, количество очков, лучшую скорость движения, средняя скорость, идентификаторы пациента, игры, типа сложности, уровня сложности, клиники, статуса архивации упражнения и измерения;

3) PATCH **/api/v1/game_results** – необходима для архивации результата игры. Принимает на вход список данных результатов игр, в которые входят: идентификатор результата игры и статус архивации. В теле ответа возвращает список идентификаторов измененных результатов игр;

л) **/api/v1/measurements** – необходима для работы с объектами результатов измерений, содержит в себе несколько методов:

1) POST **/api/v1/measurements** – необходима для создания результатов измерений. Принимает на вход список данных новых результатов измерений, в которые входят: длительность, дата создания, количество очков, лучшую скорость движения, средняя скорость, идентификаторы пациента, игры, типа сложности, уровня сложности, клиники, статуса архивации упражнения и измерения. В теле ответа возвращает список идентификаторов созданных результатов игр;

2) GET **/api/v1/measurements** – необходима для получения данных о результатах измерений. Принимает на вход необязательный параметр start_with_id, который необходим для пагинации. В теле ответа

возвращает список данных результатов игр, в которые входят: идентификатор результата измерения, дата создания и идентификаторы пациента, клиники, упражнения и статуса архивации;

3) PATCH **/api/v1/measurements** – необходима для архивации результата измерения. Принимает на вход список данных результатов измерений, в которые входят: идентификатор результата измерения и статус архивации. В теле ответа возвращает список идентификаторов измененных результатов измерений.

Функции сохранения и изменения базы данных являются одними из наиболее объемных операций в контексте работы с данными. Они обеспечивают важные механизмы для создания, обновления и хранения информации в базе данных. Для предотвращения потенциальных проблем и обеспечения надежности базы данных, часто применяется механизм транзакций. Транзакции позволяют гарантировать, что все изменения, связанные с базой данных, будут выполнены целиком и непротиворечиво, или же откатываться к предыдущему состоянию в случае возникновения ошибки (Рисунок 19).

```
async def create_session():
    session_class = get_session_class()
    session: sa_asyncio.AsyncSession = session_class()

    try:
        yield session
    except Exception as e:
        await session.rollback()
        await session.close()
        raise e
    await session.commit()
    await session.close()
```

Рисунок 19 – Откат изменений в случае возникновения ошибки

3.3 Окружение и инициализация приложения

Окружение для приложения Backend API, развертываемого с помощью Docker Compose, представляет собой комплексную систему, состоящую из нескольких контейнеров. Эти контейнеры включают базу данных PostgreSQL, систему резервного копирования wal-g и прокси-сервер nginx. Данная архитектура позволяет создать надежное и масштабируемое окружение для приложения.

Для обеспечения гибкости и настраиваемости всякий раз, когда контейнер создается, он получает настройки из файла .env. В этом файле можно указать различные параметры для каждого контейнера, такие как имя базы данных, пользовательские учетные данные, порты и другие конфигурационные параметры. Эти настройки загружаются в контейнер как переменные окружения, обеспечивая их доступность во время выполнения.

Окружение, созданное с помощью Docker Compose, также предлагает возможность работать с удаленными базами данных. Это означает, что приложение может взаимодействовать с базами данных, находящимися на других серверах или облаках, если требуется.

В целом, окружение, созданное с помощью Docker Compose и состоящее из контейнеров с базой данных PostgreSQL, системой резервного копирования wal-g и прокси-сервером nginx, предоставляет надежную и гибкую инфраструктуру для приложения Backend API. Это позволяет легко масштабировать, обеспечивать безопасность данных.

Так как в приложение контейнеризированно, всё запускается с помощью одной команды в терминале: docker-compose up -d

Инфраструктура приложения в развернутом состоянии представлена на Рисунок 20.

```
6eb27c34c0b4   backend_nginx                  sensorehab_nginx
39a545840c9c   backend_backend                sensorehab_fastapi
6d76edd27387   backend_postgres_walg          sensorehab_postgres_walg
a65c4da07f27   backend_walg                   sensorehab_walg
```

Рисунок 20 – Пример развернутой инфраструктуры

3.4 Проверка доступа к объекту

Проверка доступа к объекту происходит при отправке запроса.

Алгоритм работает следующим образом:

– сперва происходит проверка авторизационного токена на его валидность, истек ли срок его действия, а также существует ли пользователь с идентификатором, зашифрованным в токене;

– после этого проверяется, имеет ли пользователь доступ к данному методу, например, пациент не может изменять свой номер телефона напрямую, потому что он у него подтвержден;

– в последнем этапе данного алгоритма проверяется имеет ли пользователь доступ к объекту, который он пытается изменить, например, клиника не может модифицировать объекты другой клиники.

3.5 Завершение приложений

SensoRehab-Backend обрабатывает системные сигналы параллельно. Если в приложение поступает системный сигнал о прерывании, SensoRehab-Backend откатывает изменения во всех текущих сеансах работы с базой данных, останавливает все асинхронные задачи и закрывает все сеансы связи с внешними API (Рисунок 21).

```python
@app.on_event("shutdown")
async def on_shutdown_cleanup():
    adapters.scheduler.cancel_all_tasks()
    await adapters.services.external.mail.close_client()
    await adapters.clients.Httpx.close_all()
```

Рисунок 21 – Обработка системных сигналов

ЗАКЛЮЧЕНИЕ

В процессе написания выпускной квалификационной работы:

– произведен анализ предметной области и существующих на данный момент аналогов;

– разработан базовый механизм валидации входных данных;

– создано API с автоматически генерируемой интерактивной документацией;

– разработана гибкая архитектура приложения, с помощью которой возможно вертикальное масштабированное;

– достигнута возможность горизонтального масштабирования, благодаря правильной архитектуре базы данных и соответствующей логике, что позволяет обработку запросов на нескольких экземплярах SensoRehab-Backend;

– описаны структуры, соответствующие моделям базы данных;

– интегрирован сервис отслеживания ошибок Sentry, который отслеживает необработанные ошибки в приложении SensoRehab-Backend и отправляет уведомления об ошибках в выбранный канал связи (в данном случае, реализован только slack);

– был создан Docker-образ для приложения. Данные для доступа к базе данных, токены авторизации к сторонним сервисам и другие переменные сохраняются внутри контейнера как переменные окружения.

В результате были достигнуты все поставленные задачи, и получен Акт о внедрении (Приложение А) в производственный процесс компании «Сенсомед». На данный момент, приложение полностью используется компанией.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Данные по инсультам в Красноярском Крае [Электронный ресурс]. – URL: https://ngs24.ru/text/health/2020/10/29/69521501 (дата обращения: 17.11.2022).

2. Инсульты в России [Электронный ресурс]. – URL: https://cyberleninka.ru/article/n/insult (дата обращения: 17.11.2022).

3. Реабилитационная перчатка Аника [Электронный ресурс]. – URL: https://www.madin.ru/catalog/direction/oborudovanie-dlya-manualnoy-terapii/reabilitatsionnaya-perchatka-anika/ (дата обращения: 12.12.2022).

4. Реабилитационная перчатка Rafael [Электронный ресурс]. – URL: https://www.neofect.com/us/smart-glove (дата обращения: 12.12.2022).

5. Реабилитационная перчатка Hand Tutor [Электронный ресурс]. – URL: https://meditouch.co.il/products/handtutor/ (дата обращения: 12.12.2022).

6. Система контейнерной виртуализации Docker [Электронный ресурс]. – URL: https://www.docker.com (дата обращения: 09.02.2023).

7. Приложение для отслеживания ошибок Sentry [Электронный ресурс]. – URL: https://sentry.io (дата обращения: 14.03.2023).

8. Веб-фреймворк FastAPI [Электронный ресурс]. – URL: https://fastapi.tiangolo.com (дата обращения: 08.04.2023).

9. База данных PostgreSQL [Электронный ресурс]. – URL: https://www.postgresql.org (дата обращения: 08.04.2022).

10. Стандарт университета «Общие требования к построению, изложению и оформлению документов учебной деятельности» [Электронный ресурс]. – URL: https://about.sfu-kras.ru/docs/8127/pdf/791393 (дата обращения: 17.11.2022).

ПРИЛОЖЕНИЕ А. Акт о внедрении

Министерство науки и высшего образования РФ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Кафедра вычислительной техники
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О.В.Непомнящий
подпись инициалы, фамилия
«22» 06 2023г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – Информатика и вычислительная техника
код – наименование направления

Backend API для приложения SensoRehab компании «Sensomed»
тема

Руководитель	22.06.23	доц, к.т.н	Л. И. Покидышева
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	22.06.2023		Т. О. Салтымаков
	подпись, дата		инициалы, фамилия
Нормоконтролер	22.06.23		Л. И. Покидышева
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2023