

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования

**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

« ____ » _____ 2023 г.

Разработка веб-приложения для студенческого совета ЮУрГУ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 09.03.04.2023.308-286.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.

_____ В.Н. Алеева

Автор работы,
студент группы КЭ-403

_____ Д.В. Полуротов

Ученый секретарь
(нормоконтролер)

_____ И.Д. Володченко

« ____ » _____ 2023 г.

Челябинск, 2023 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования

**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук
Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

_____ Л.Б. Соколинский

06.02.2023 г.

ЗАДАНИЕ

на выполнение выпускной квалификационной работы бакалавра

студенту группы КЭ-403

Полуротову Денису Владимировичу,

обучающемуся по направлению

09.03.04 «Программная инженерия»

1. Тема работы (утверждена приказом ректора от 25.04.2023 г. №753-13/12)
Разработка веб-приложения для студенческого совета ЮУрГУ.

2. Срок сдачи студентом законченной работы: 05.06.2023 г.

3. Исходные данные к работе

3.1. Руководство по ASP.NET Core 7. [Электронный ресурс] URL:

<https://metanit.com/sharp/aspnet6/> (дата обращения: 14.02.2023 г.).

3.2. Учебник. Начало работы с ASP.NET Core. [Электронный ресурс] URL:

<https://learn.microsoft.com/ru-ru/aspnet/core/getting-started/?view=aspnetcore-7.0&tabs=windows> (дата обращения: 14.02.2023 г.).

4. Перечень подлежащих разработке вопросов

4.1. Создание базы данных.

4.2. Создание API.

4.3. Создание клиента.

4.4. Подготовка документации для реализованной системы.

4.5. Опытное внедрение системы.

5. Дата выдачи задания: 06.02.2023 г.

Научный руководитель,

доцент кафедры СП, к.ф.-м.н.

В.Н. Алеева

Задание принял к исполнению

Д.В. Полуротов

ГЛОССАРИЙ

1. *Алгоритм SHA-256* – представляет собой алгоритм, который принимает в качестве входных данных любые произвольные данные и создает серию уникальных цифр и букв [1].

2. *Десктопное приложение* – программа, которая устанавливается на компьютер пользователя и работает под управлением операционной системы [2].

3. *Фреймворк* – готовая модель в IT, заготовка, шаблон для программной платформы, на основе которого можно дописать собственный код [3].

4. *API (Application programming interface)* – это механизмы, которые позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов [4].

5. *Opensource* – это децентрализованная модель разработки, которая позволяет любому человеку изменять технологию и обмениваться ею, поскольку ее структура находится в открытом доступе [5].

6. *Транзакция* – это набор операций по работе с базой данных (БД), объединенных в одну атомарную пачку [6].

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ.....	3
ВВЕДЕНИЕ.....	5
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	7
1.1. Описание предметной области	7
1.2. Сравнительный анализ аналогов	7
2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	9
2.1. Платформа ASP.NET	9
2.2. REST API.....	10
2.3. Реляционные базы данных	11
2.4. Docker	12
3. ПРОЕКТИРОВАНИЕ	14
3.1. Анализ требований к программной системе	14
3.2. Диаграмма вариантов использования	14
3.3. Диаграмма компонентов	18
4. РЕАЛИЗАЦИЯ.	19
4.1. Средства разработки	19
4.2. Создание схемы базы данных.....	19
4.3. Создание таблиц базы данных.....	20
4.4. Создание API	21
4.5. Создание контроллеров	22
4.6. Развертывание в Docker Container.....	23
4.7. Создание клиента	26
5. ТЕСТИРОВАНИЕ СИСТЕМЫ.....	34
5.1. Функциональное тестирование.....	34
ЗАКЛЮЧЕНИЕ	36
ЛИТЕРАТУРА.....	37
ПРИЛОЖЕНИЕ. Спецификация вариантов использования.....	40

ВВЕДЕНИЕ

Актуальность

Каждые уважающие себя институты и высшие школы стараются улучшить жизнь своих студентов различными способами. Один из таких способов – создание студенческих советов. В нашем университете их множество, и каждый работает так, как ему кажется лучше. Но не все институты и высшие школы справляются с тем объемом задач, который ложится на них. Причина проста – плохая организованность процессов. Чтобы улучшить свою работу, некоторые высшие школы и институты стараются цифровизировать свою работу. Студенческий совет ВШЭКН в этом плане выступает лидером среди других студенческих советов. Мною, как давнему участнику студенческого совета, была предложена идея создания общей базы для всех студенческих советов, которая бы объединила и стандартизировала бы работу всех студенческих советов ЮУрГУ. Эта инициатива была поддержана большинством председателей других студенческих советов (опрос проводился в личных беседах). Эта база должна представлять из себя несколько модулей, написанных на разных платформах для удобного использования.

Если говорить об эффективности подобных решений, то в истории немало таких примеров. Все они отвечают одному главному критерию, собрание всей информации в одном месте, с быстрым доступом к ней. Неспроста люди начали прибегать к информационным системам еще в древности, чтобы точно знать, сколько еды у них имеется. Это позволяет управлять большим количеством людей и материальных благ практически без потерь. Первые подобные системы появились еще у древних шумер, где во время урожайного года люди несли свои припасы в храмы, где специальный человек ставил черточку каждый раз, когда мимо него проходил человек с зерном. Это было нужно для того, чтобы понимать, сколько сейчас находится припасов в храме. Помимо простых черточек, сверху рисо-

вался значок того ресурса, который заносился в храм. Таким образом всегда можно было примерно знать, сколько того или иного ресурса сейчас находится в запасах у города. А когда настанет голодный год, то гражданам будет выдана такая часть, чтобы запасы не иссякли, но горожане не погибли от голода.

В наше время подобных систем масса. В век цифровизации люди стали строить гораздо более сложные системы, чем те, что были в древности, но их смысл всегда остается один – контроль, порядок и точность. Все современные системы соблюдают эти принципы, поскольку от них зависит работа целых производств.

Постановка задачи

Целью выпускной квалификационной работы является разработка API и базы данных для приложения студенческих советов ЮУрГУ. Для достижения поставленной цели необходимо решить следующие задачи:

- 1) спроектировать и создать базу данных;
- 2) разработать API, для управления клиентской частью проекта;
- 3) разработать клиента;
- 4) внедрить систему в работу студенческих советов.

Структура и содержание работы

Работа состоит из введения, 5 глав, заключения и списка литературы. Объем работы составляет 43 страницы, объем списка литературы – 25 источников.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

В этой работе рассматривается вопрос создания веб-приложения для студенческих советов ЮУрГУ. Это необходимо для роста программного продукта, поскольку в современном мире все изменяется очень быстро, и заставлять пользователей обновлять программу практически ежедневно невозможно. Именно поэтому было принято решение сделать систему из трех уровней: клиент, API, база данных. Это позволит обновлять программу без вмешательства в дела пользователей, лишь изредка менять клиентскую часть, в результате глобальных обновлений.

В качестве платформы для разработки была выбрана технология ASP.NET. Эта платформа берет свое начало с 2002 года, и развивается до сих пор [7]. Сейчас она именуется как ASP.NET Core, версия 7. Именно она и будет использоваться в проекте.

1.2. Сравнительный анализ аналогов

В настоящее время не существует полноценных программ для работы студенческих советов. На замену им можно предложить различные HRM. HRM (Human Resources Management) – это системы для управления человеческими ресурсами, проще говоря, для управления людьми. В этих программах можно видеть весь свой персонал, задачи каждого сотрудника, его заработную плату. Как пример – программа БОСС-Кадровик. Основное назначение АРМ БОСС-Кадровик - учет и управление персоналом, ведение организационной и штатной структуры компании, учет рабочего времени, расчет заработной платы. В состав системы БОСС-Кадровик включены следующие функциональные модули:

- 1) штатное расписание;
- 2) учет кадров;
- 3) табельный учет;

- 4) расчет заработной платы;
- 5) учет для Пенсионного фонда России;
- 6) администратор системы.

Недостатком данной системы является ее дороговизна, поэтому она точно не подходит для организации процессов в студенческом совете. Пример интерфейса представлен на рисунке 1.

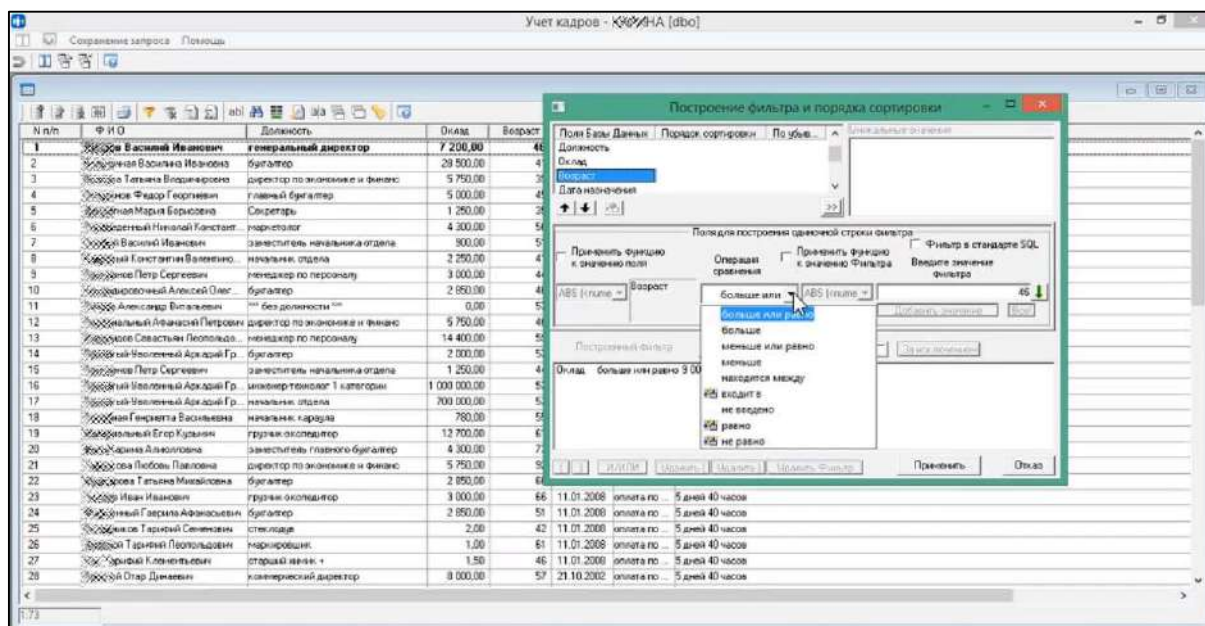


Рисунок 1 – HRM БОСС-Кадровик

Вывод по первой главе

По итогам анализа предметной области были сформулированы особенности, на которые необходимо будет обратить внимание при разработке.

1. Система должна обладать гибкостью и надежностью.
2. Система должна быть спроектирована так, чтобы в будущем ее было легко поддерживать.
3. Система должна легко расширяться.
4. В системе должны быть описаны все области работы целевой аудитории.

2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1. Платформа ASP.NET

ASP.NET Core – это opensource-фреймворк, все его файлы доступны на GitHub. Архитектура платформы представлена на рисунке 2.

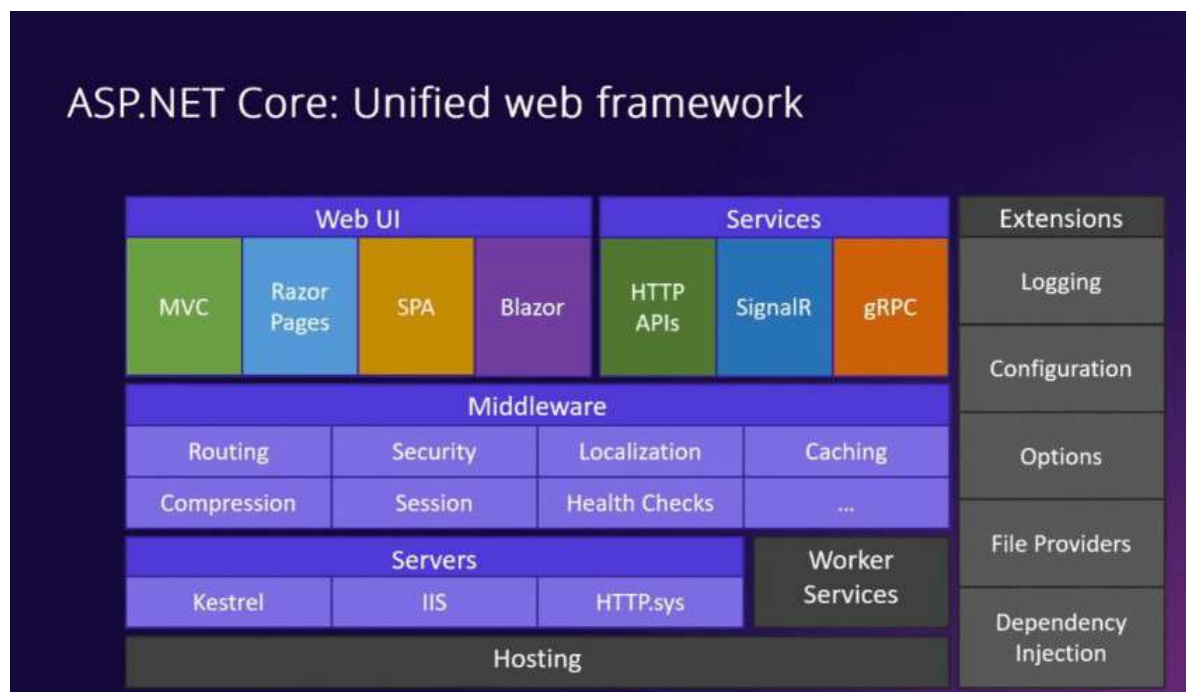


Рисунок 2 – Архитектура платформы ASP.NET Core

На самом верхнем уровне располагаются различные модели взаимодействия с пользователем [7]. Это технологии построения пользовательского интерфейса и обработки ввода пользователя, как MVC, Razor Pages, SPA (Single Page Application – одностраничные приложения с использованием Angular, React, Vue) и Balzor. Кроме того, это сервисы в виде встроенных HTTP API, библиотеки SignalR или сервисов GRPC.

Все эти технологии базируются и/или взаимодействуют с чистым ASP.NET Core, который представлен прежде всего различными встроенными компонентами middleware-компонентами, которые применяются для обработки запроса. Кроме того, технологии высшего уровня также взаимодействуют с различными расширениями, которые не являются непосред-

ственной частью ASP.NET Core, как расширения для логгирования, конфигурации и т.д.

И на самом нижнем уровне приложение ASP.NET Core работает в рамках некоторого веб-сервера, например, Kestrel, IIS, библиотеки HTTP.sys, или при помощи программного обеспечения Docker.

В нашей работе будет использоваться MVC в качестве паттерна при проектировании приложения. Также, в качестве среды развертывания будет использовано программное обеспечение Docker.

2.2. REST API

REST API – это такой вид API, в котором клиент отправляет запросы на сервер в виде данных, а сервер использует эти данные для запуска внутренних функций и возвращает выходные данные обратно клиенту [8].

Архитектура REST имеет несколько принципов [4].

1. Единый интерфейс – это конструктивная основа любого веб-сервиса REST. Это означает то, что сервер передает информацию в стандартном формате. Формат может отличаться от внутреннего ресурса в серверном приложении. К примеру, сервер хранит данные в виде текста, но передавать он будет их в виде JSON.

2. Отсутствие сохранения состояния. Это относится к методу связи, при котором сервер выполняет каждый запрос независимо от всех предыдущих запросов. Это конструктивное ограничение подразумевает, что сервер может каждый раз полностью понять и выполнить запрос.

3. Многоуровневая система. В многоуровневой системной архитектуре клиент может подключаться к другим авторизованным посредникам между клиентом и сервером и по-прежнему получать ответы от сервера. Серверы также могут передавать запросы другим серверам. Вы можете спроектировать свою веб-службу REST для работы на нескольких серверах с несколькими уровнями (безопасностью, приложениями и бизнес-

логикой), совместно выполняющих клиентские запросы. Эти уровни остаются невидимыми для клиента.

4. Кэширование. Все веб-службы поддерживают кэширование, то есть процесс сохранения некоторых ответов на клиенте или на посреднике для увеличения скорости ответа сервера. К примеру, если вы знаете, что список факультетов в университете меняется очень редко, то есть смысл закэшировать эти данные с первого запроса, и в последствии пользоваться ими, уменьшая время ответа сервера.

5. Код по запросу. В архитектурном стиле REST серверы могут временно расширять или настраивать функциональные возможности клиента, передавая код программного обеспечения. Например, когда вы заполняете регистрационную форму на каком-либо веб-сайте, ваш браузер сразу же выделяет все допущенные ошибки (например, неверные номера телефонов). Это происходит благодаря коду, отправленному сервером.

2.3. Реляционные базы данных

В основе любой информационной системы лежит база данных, которая описывает все сущности системы. В древности это были глиняные пластины, после их сменил папирус, а ближе к нашему времени его заменили книги. Сейчас нет необходимости в большом количестве бумажных носителей. Вместо них существуют цифровые базы данных. Сейчас существует несколько типов баз данных.

1. Реляционные базы данных. Реляционные базы данных стали преобладать в 1980-х годах. Данные в реляционной базе организованы в виде таблиц, состоящих из столбцов и строк. Реляционная СУБД обеспечивает быстрый и эффективный доступ к структурированной информации [9].

2. Объектно-ориентированные базы данных. Это такая база данных, в которой данные оформлены в виде моделей объектов, включающих прикладные программы, которые управляются внешними событиями [10].

3. Распределенные базы данных. Распределенная база данных – это такой набор отношений, хранящихся в разных узлах компьютерной сети и логически связанных таким образом, чтобы составлять единую совокупность данных [11].

4. Хранилища данных. Это разновидность системы управления данными, которая обеспечивает поддержку бизнес-аналитики. Хранилища данных предназначены только для выполнения запросов и анализа и обычно содержат большие объемы исторических данных. Данные обычно поступают в хранилище из самых различных источников, таких как журналы приложений и приложения транзакций [12].

Для информационных систем лучше всего подходят реляционные и объектно-реляционные базы данных, такие как MS SQL Server и PostgreSQL. Такие виды баз данных лучше всего подходят потому, что они обеспечивают лучшую производительность и самый быстрый доступ. Так же они наиболее удобны в применении как программисту, так и обычному пользователю.

2.4. Docker

Прежде чем рассказывать про Docker, нужно сказать несколько слов о технологии контейнеризации [13].

Контейнеры – это способ стандартизации развертки приложения и отделения его от общей инфраструктуры. Экземпляр приложения запускается в изолированной среде, не влияющей на основную операционную систему.

Разработчикам не нужно задумываться, в каком окружении будет работать их приложение, будут ли там нужные настройки и зависимости. Они просто создают приложение, упаковывают все зависимости и настройки в некоторый единый образ. Затем этот образ можно запускать на других системах, не беспокоясь, что приложение не запустится.

Docker – это платформа для разработки, доставки и запуска контейнерных приложений. Docker позволяет создавать контейнеры, автоматизировать их запуск и развертывание, управляет жизненным циклом. Он позволяет запускать множество контейнеров на одной хост-машине.

Контейнеризация похожа на виртуализацию, но это не одно и то же. Виртуализация запускает полноценный хост на гипервизоре со своим виртуальным оборудованием и операционной системой. При этом внутри одной ОС можно запустить другую ОС. В случае контейнеризации процесс запускается прямо из ядра основной операционной системы и не виртуализует оборудование. Это означает, что контейнеризованное приложение может работать только в той же ОС, что и основная. Контейнеры не виртуализируют оборудование, поэтому потребляют меньше ресурсов.

Вывод по второй главе

Технологии, выбранные для реализации проекта, одновременно надежные и современные. Эти технологии достаточно популярны, чтобы этот проект мог развиваться и поддерживаться без проблем с поиском программистов, разбирающихся в этих технологиях. Вместе с тем, эти технологии самодостаточны, поэтому система получится стрессоустойчивой.

3. ПРОЕКТИРОВАНИЕ

3.1. Анализ требований к программной системе

В ходе проектирования были определены требования к системе и возможности, которые должны предоставляться пользователю при использовании системы.

Функциональные требования – это условия, которые накладываются на поведение системы, а также действия, которые система имеет возможность выполнять.

Разрабатываемая система должна удовлетворять следующим функциональным требованиям:

- 1) система должна принимать на вход запросы в виде JSON;
- 2) система должна отправлять ответ в виде JSON с указанием кода HTTP.

Нефункциональные требования – это условия, которые не связаны с поведением системы и характеризуют условия окружающей среды или же качества, которыми должна обладать система.

Разрабатываемая система должна удовлетворять следующим нефункциональным требованиям:

- 1) система должна быть реализована на платформе ASP.NET, на языке C#;
- 2) база данных должна быть реализована на SQL с использованием СУБД MSSQL;
- 3) система должна быть реализована таким образом, чтобы в будущем было возможно использовать любой тип клиента.

3.2. Диаграмма вариантов использования

В ходе анализа требований к системе была разработана UML-диаграмма вариантов использования [14], представленная на рисунке 3.

В данной диаграмме есть несколько актеров, взаимодействующих с системой, это Участник студсовета, Зам. председателя студсовета, Председатель студсовета и Администратор.

Для данных актеров было спроектировано несколько вариантов использования:

- 1) посмотреть состав студсовета;
- 2) посмотреть инвентарь студсовета;
- 3) исправить информацию о человеке из студсовета;
- 4) исправить информацию об инвентаре;
- 5) создать нового члена студенческого совета;
- 6) создать новую запись единицы инвентаря;
- 7) изменить пароль пользователя и изменить данные пользователя.

Спецификации вариантов использования описаны в приложении А.

Рассмотрим подробнее варианты использования, определенные для пользователей. Все пользователи, кроме актера «Администратор», являются участниками студсовета, поэтому наследуются от актера «Участник студсовета».

Рассмотрим варианты использования для актера «Участник студсовета»:

- 1) посмотреть состав студсовета – пользователь получает список всех участников своего студенческого совета, но те участники, что находятся в архиве, изначально скрыты;
- 2) посмотреть инвентарь студсовета – пользователь получает список инвентаря студенческого совета, но те позиции, что находятся в архиве, изначально скрыты;
- 3) изменить свои данные – пользователь может изменить свой собственный пароль, введя свой настоящий для подтверждения.

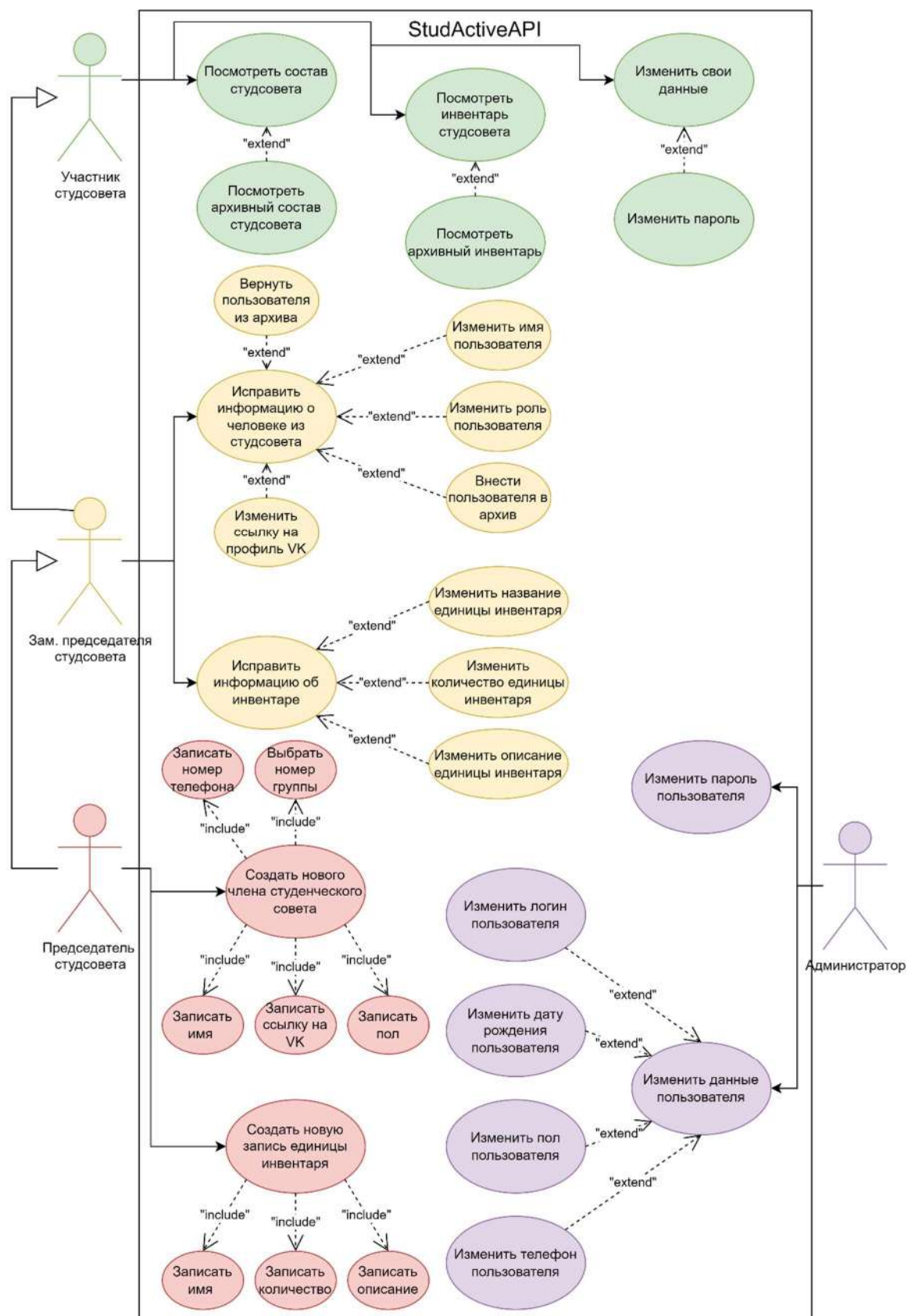


Рисунок 3 – Диаграмма вариантов использования

Далее рассмотрим варианты использования для актера «Зам. Председателя студсовета»:

1) исправить информацию о человеке из студсовета – пользователь может исправить некоторую информацию о человеке из студсовета, к примеру, вернуть его из архива, или наоборот, внести в архив, изменить имя данного пользователя, изменить его роль и так далее;

2) исправить информацию об инвентаре – пользователь может исправить различную информацию об единице инвентаря, к примеру, увеличить или уменьшить количество на балансе студенческого совета, изменить описание инвентаря, имя инвентаря.

Теперь рассмотрим варианты использования для актера «Председатель студсовета»:

1) создать нового члена студенческого совета – для создания нового члена студсовета пользователь должен заполнить данные нового члена, такие как имя, ссылка на VK, номер телефона, пол, имя, номер группы;

2) создать новую запись единицы инвентаря – для создания новой записи пользователь должен заполнить данные единицы инвентаря, такие как имя инвентаря, описание и количество.

И наконец, рассмотрим варианты использования актера «Администратор»:

1) изменить пароль пользователя – администратор может изменить актуальный пароль пользователя, в случае, если пользователь забыл свой актуальный пароль;

2) изменить данные пользователя – администратор может изменить данные пользователя, такие как логин, дата рождения, пол, телефон.

3.3. Диаграмма компонентов

Основными компонентами системы являются клиент, API и база данных.

Клиент – необходим для взаимодействия пользователя и API. Общение происходит при помощи REST-запросов.

API – принимает данные от клиентов и обрабатывает их в зависимости от полученных данных. Отправляет ответы в соответствии с кодами HTTP.

База данных – хранит данные в таблицах, поступающие от API.

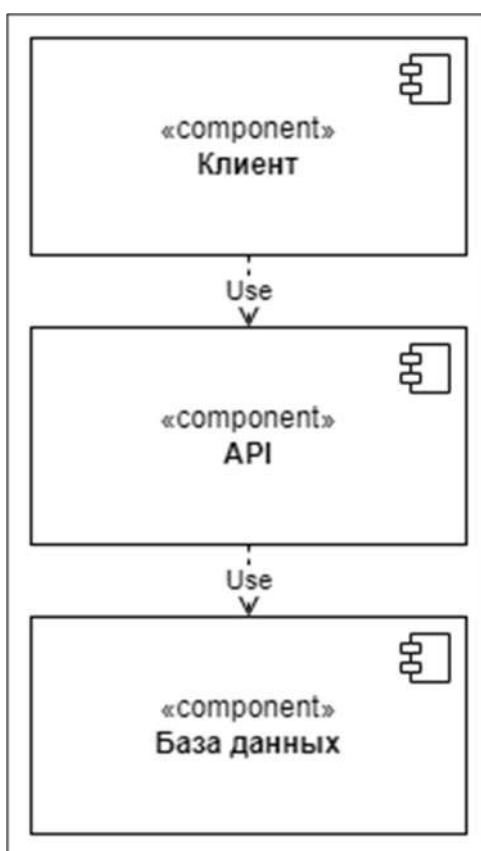


Рисунок 4 – Диаграмма компонентов системы

Вывод по третьей главе

В ходе проектирования были описаны функциональные и нефункциональные требования, которым должна удовлетворять система, а также спроектирована диаграмма вариантов использования и диаграмма компонентов системы.

4. РЕАЛИЗАЦИЯ.

4.1. Средства разработки

В качестве основного языка разработки программной части был выбран язык C# версии 11 [15], поскольку этот язык является довольно популярным, вследствие чего поддержка системы будет проще. Данное приложение в будущем планируется расширять под нужды студенческих советов, а также разработка этого приложения будет являться хорошей практикой для студентов Высшей Школы Электроники и Компьютерных Наук. Версия языка является новейшей, что будет актуально для молодых разработчиков, которые получают опыт разработки на современной версии языка.

Также для языка была выбрана платформа .NET Core 7.0, которая на момент 2023 года является новейшей [16]. В .NET Core 7.0 главным преимуществом является то, что в эту версию была добавлена улучшенная поддержка платформы Linux, что позволит быстрее разворачивать контейнеры.

Средой разработки будет выступать Visual Studio 2022. Эта версия получила усовершенствованный IntelliCode, что помогает в разработке и ускоряет написание кода, и предотвращает ошибки.

4.2. Создание схемы базы данных

Перед тем, как открыть приложение для создания базы данных, необходимо спроектировать схему этой базы данных. Это нужно, чтобы сразу выявить различные недостатки системы, и не исправлять их в уже созданных таблицах, делая двойную работу [17]. Схема базы данных для приложения студенческих советов ЮУрГУ представлена на рисунке 5.

По данной схеме необходимо будет создать таблицы, вместе с указанными связями.

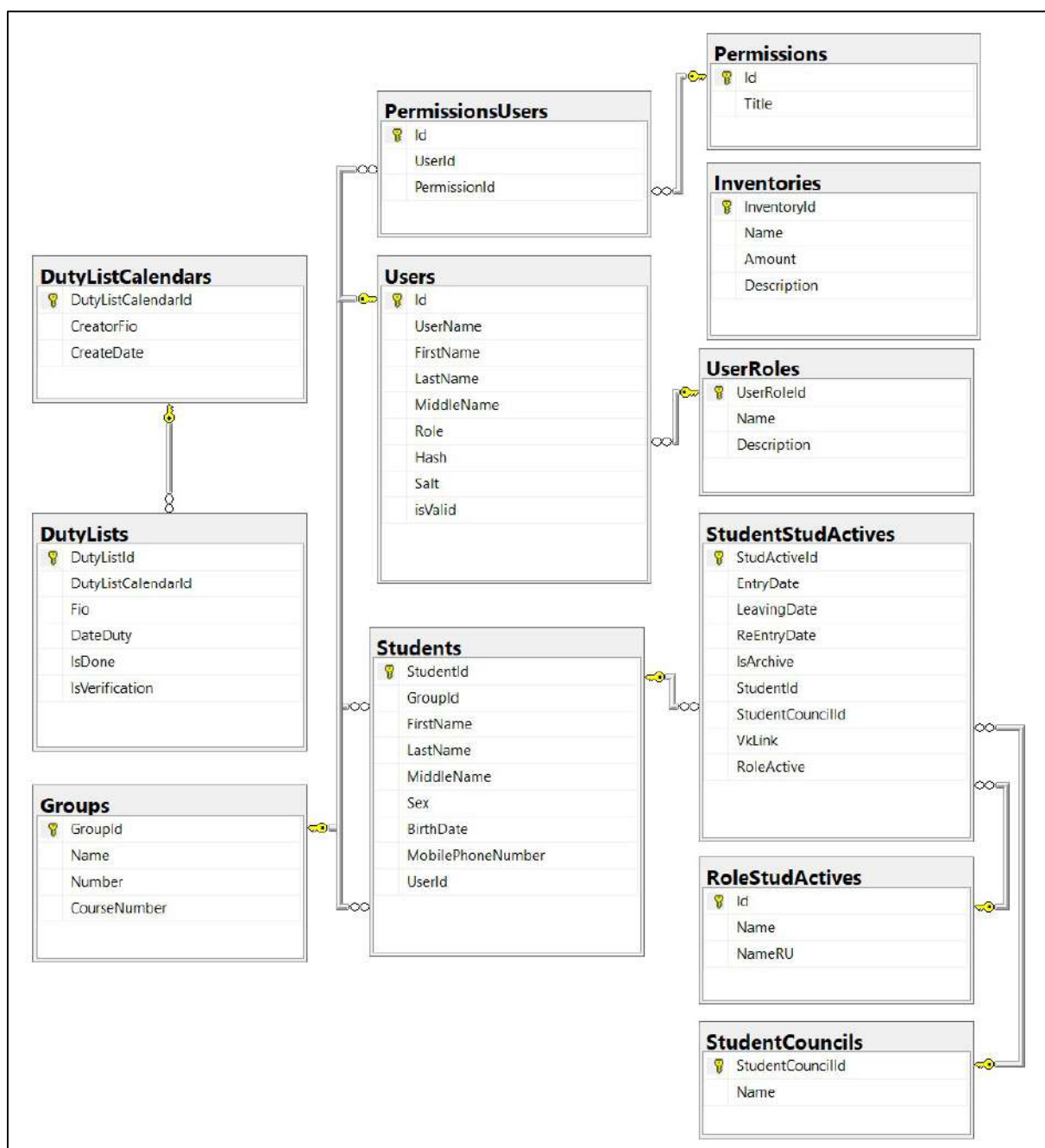


Рисунок 5 – Схема базы данных

4.3. Создание таблиц базы данных

Для разработки базы данных была выбрана СУБД MSSQL SMS. В этой СУБД используется язык SQL, язык структурированных запросов. В MSSQL есть возможность создания таблиц при помощи интерфейса, но также возможно создание таблиц при помощи SQL-запросов [18]. К при-

меру, для создания таблицы Users был использован код, представленный на рисунке 6.

```
CREATE TABLE [dbo].[Users](
    [Id] [uniqueidentifier] NOT NULL,
    [UserName] [nvarchar](max) NOT NULL,
    [FirstName] [nvarchar](max) NULL,
    [LastName] [nvarchar](max) NULL,
    [MiddleName] [nvarchar](max) NULL,
    [Role] [uniqueidentifier] NULL,
    [Hash] [nvarchar](max) NULL,
    [Salt] [varbinary](max) NULL,
    CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
(
    [Id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTI-
MIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT
[FK_Users_UserRoles] FOREIGN KEY([Role])
REFERENCES [dbo].[UserRoles] ([UserRoleId])
GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK_Users_UserRoles]
GO
```

Рисунок 6 – Код, создающий таблицу Users в СУБД MSSQL

После создания таблиц, необходимо их заполнить данными. Можно воспользоваться интерфейсом СУБД, или заполнить их после реализации системы уже в интерфейсе клиента.

4.4. Создание API

Для создания API была выбрана платформа ASP.NET. Чтобы создать API на данной платформе, необходимо установить среду разработки Visual Studio. После установки и создания пустого проекта API, необходимо сразу перейти в файл Program.cs и изменить его согласно рисунку 7 [19].

Этот файл является самым главным в проекте API. С него начинается запуск всего проекта. После этого мы можем приступить к созданию контроллеров.

```

using Microsoft.EntityFrameworkCore;
using StudActiveAPI.Services;

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllers();
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
var connectionString = builder.Configuration.GetConnectionString("ApiAppCon");
builder.Services.AddDbContext<Context>(op =>
op.UseSqlServer(connectionString));
var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseRouting();
app.UseWhen(context => context.Request.Path.StartsWithSegments("/api/event") || context.Request.Path.StartsWithSegments("/api/device"), appBuilder => {
appBuilder.UseMiddleware<Middleware>(); });
app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();

```

Рисунок 7 – Файл Program.cs

4.5. Создание контроллеров

Контроллеры выполняют функцию некоего «входа в программу». При помощи REST запросов мы можем управлять этим API [20]. К примеру, код на рисунке 8 демонстрирует контроллер, благодаря которому мы можем получить список всех пользователей этой системы.

```

using Microsoft.AspNetCore.Mvc;
using StudActiveAPI.Models;
using StudActiveAPI.Services;

namespace StudActiveAPI.Controllers
{
    [ApiController]
    [Route("api/[controller]")]
    public class UserController : Controller
    {
        private Context _context;
        public UserController(Context context) { _context = context; }
        [HttpGet]
        public JsonResult GetUsers()
        {
            return new JsonResult(_context.Users.ToList());
        }
    }
}

```

Рисунок 8 – Код контроллера UserController

Тип возвращаемых данных всегда будет в виде JSON. Это необходимо для гибкости программы. К примеру, к этой API в будущем планируется создать три клиента – десктопное приложение, мобильное приложение и вебсайт, это необходимо для удобного пользования всем пользователям, вне зависимости от их предпочтений в пользовании системы. Для всех этих клиентов может использоваться совершенно разная архитектура, и, чтобы все эти приложения могли общаться, необходим единый вид сообщений. Поэтому тип JSON был выбран как единый для всех [21].

4.6. Развертывание в Docker Container

Для развертывания проекта .NET в Docker Container для начала необходимо установить приложение Docker Desktop. Интерфейс представлен на рисунке 9.

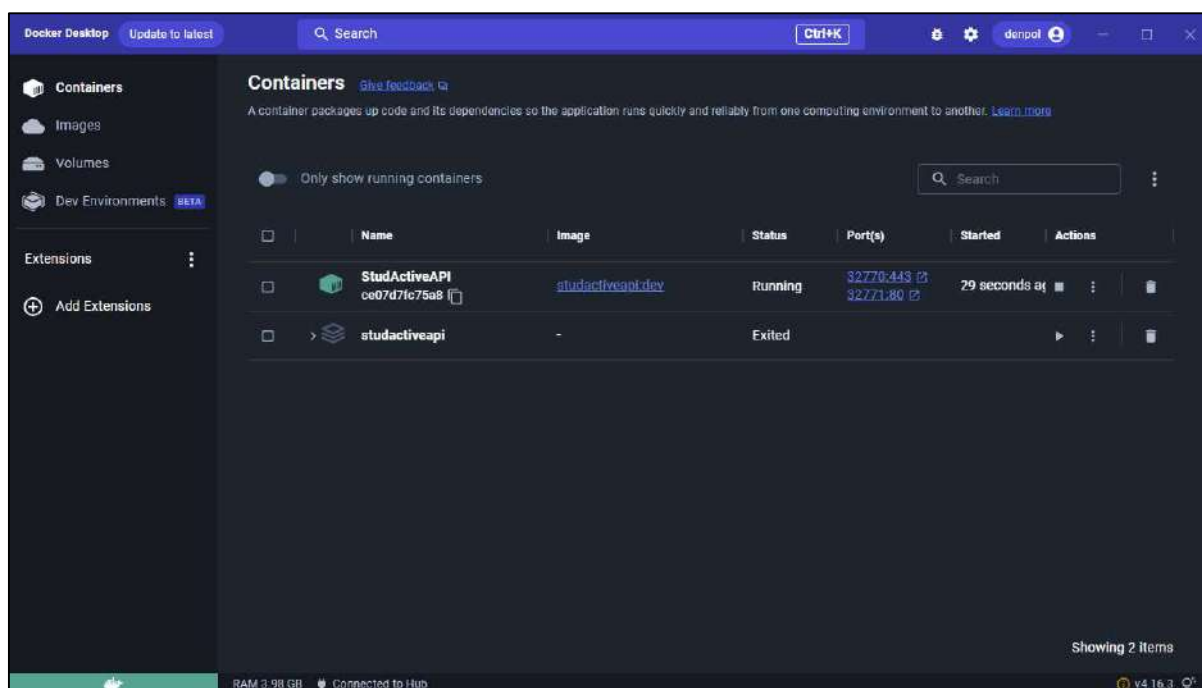


Рисунок 9 – Docker Desktop

На рисунке 8 можно увидеть запущенные контейнеры. В нашем случае это контейнер с базой данных и API под названием studactiveapi. Для создания контейнера сначала необходимо написать так называемый докер-файл [22-23], код которого представлен на рисунке 10. После создания докер-файла и запуска API необходимо написать файл docker-compose.yml [23-24]. Код данного файла представлен на рисунке 11. Этот файл необходим для создания контейнера, в котором мы развернем помимо нашей API еще и базу данных. Это необходимо для качественного администрирования серверной части нашего приложения. После запуска API с этим файлом докер создаст контейнер с API и базой данных, как это видно на рисунке 12. API будет доступна по порту 7777:443, а база данных по порту 7778:1433.


```

FROM mcr.microsoft.com/dotnet/aspnet:7.0 AS base
WORKDIR /app
EXPOSE 80
EXPOSE 443

FROM mcr.microsoft.com/dotnet/sdk:7.0 AS build
WORKDIR /src
COPY ["StudActiveAPI.csproj", "StudActiveAPI/"]
RUN dotnet restore "StudActiveAPI/StudActiveAPI.csproj"

WORKDIR "/src/StudActiveAPI"
COPY . .

RUN dotnet build "StudActiveAPI.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "StudActiveAPI.csproj" -c Release -o /app/publish
/p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "StudActiveAPI.dll"]

```

Рисунок 10 – Код докер-файла

```

version: '3.8'

services:
  run-api-locally:
    depends_on:
      - stud-active-api
    image: tianon/true
    restart: "no"

  stud-active-api:
    container_name: stud-active-api
    build:
      context: .
    ports:
      - "7777:443"
    depends_on:
      - stud-active-db

  stud-active-db:
    container_name: stud-active-db
    image: 'mcr.microsoft.com/mssql/server'
    ports:
      - '7778:1433'
    environment:
      - ACCEPT_EULA=Y
      - MSSQL_SA_PASSWORD=1YX3aE0Q57
    volumes:
      - './drive:/var/opt/mssql'

```

Рисунок 11 – Код файла docker-compose.yml

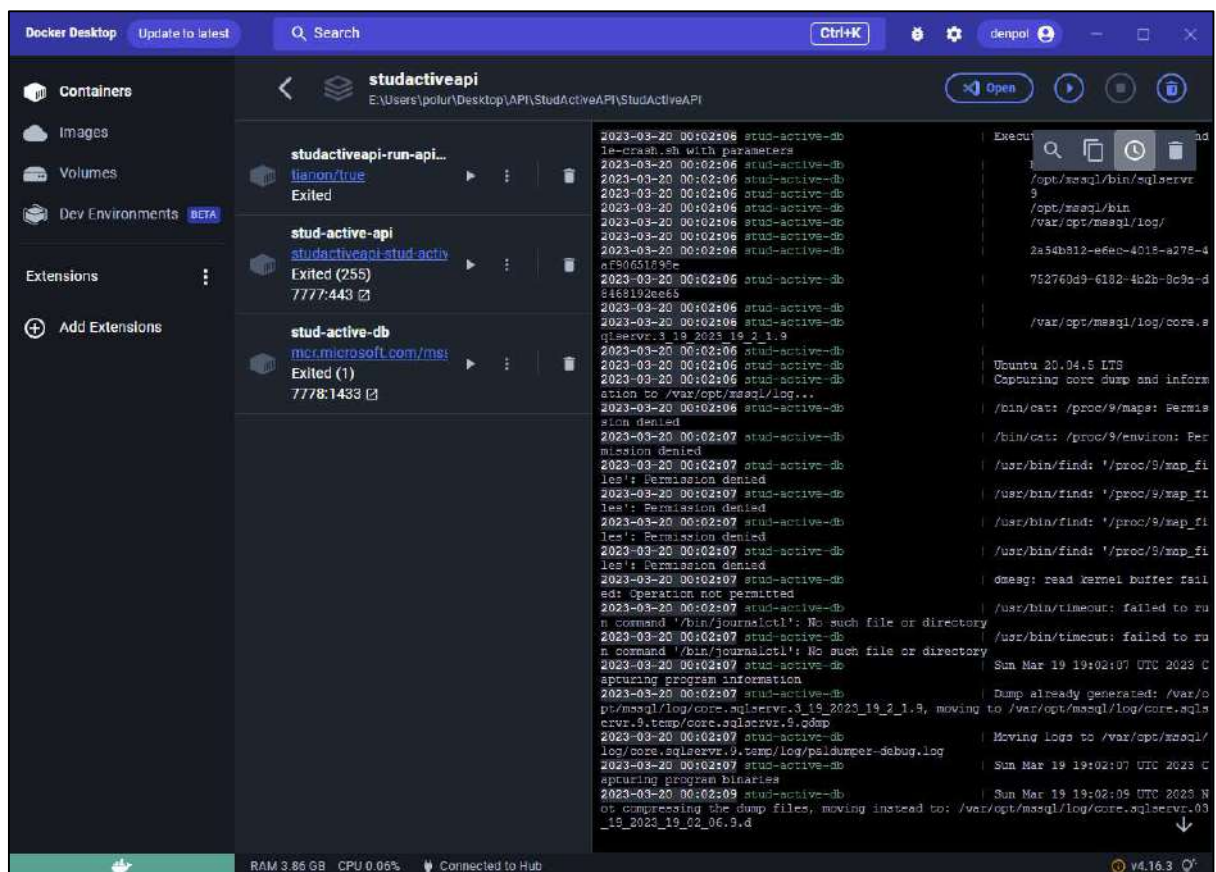


Рисунок 12 – Контейнер с API и базой данных

4.7. Создание клиента

Для того, чтобы пользователь мог работать в системе, ему необходим клиент. В нашем случае клиентом будет выступать десктопное приложение, созданное на платформе .NET с использованием технологии WPF.

Первое, что встречает пользователя, это окно входа (рисунок 13). На нем мы можем увидеть два поля – поле логина и поле пароля. В них вводится соответствующая информация. После нажатия на кнопку «Войти» программа отправляет запрос на API для проверки данных, и в случае подтверждения личности пропускает пользователя в систему с соответствующими правами.

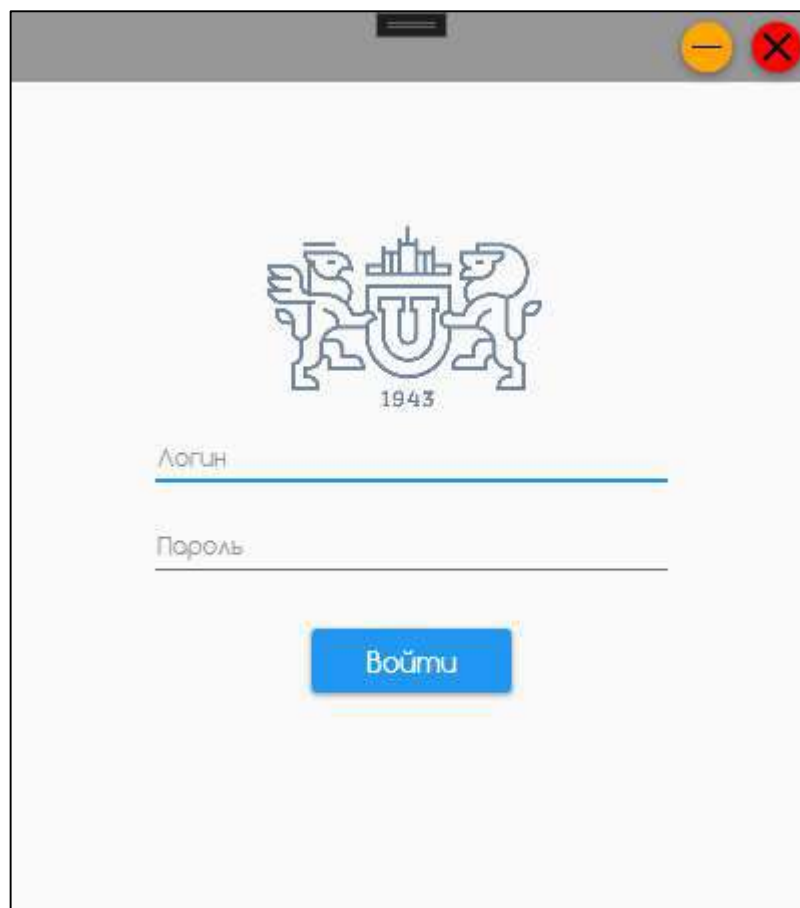


Рисунок 13 – Окно входа в систему

После успешного входа пользователь попадает на страницу личного кабинета (рисунок 14). В данной странице пользователь может посмотреть свои данные, изменить свой пароль и перейти на свою страницу ВК.

В левом верхнем углу находится иконка раскрывающегося меню. После нажатия на нее, открывается меню, в котором пользователь может переходить в другие разделы (рисунок 15). В этом же меню можно изменить тему приложения на темную, изменив положение переключателя в конце списка разделов (рисунок 16). Для удобства отображения рисунков, все дальнейшие демонстрации разделов будут выполнены в режиме светлой темы.

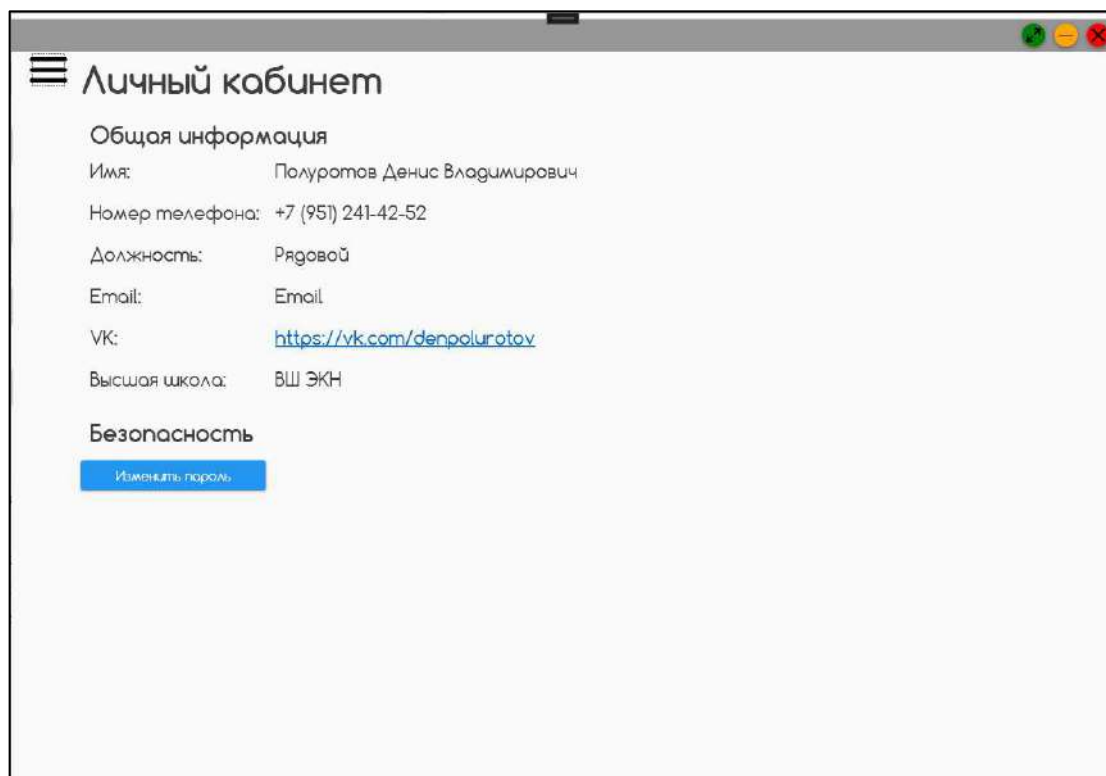


Рисунок 14 – Личный кабинет

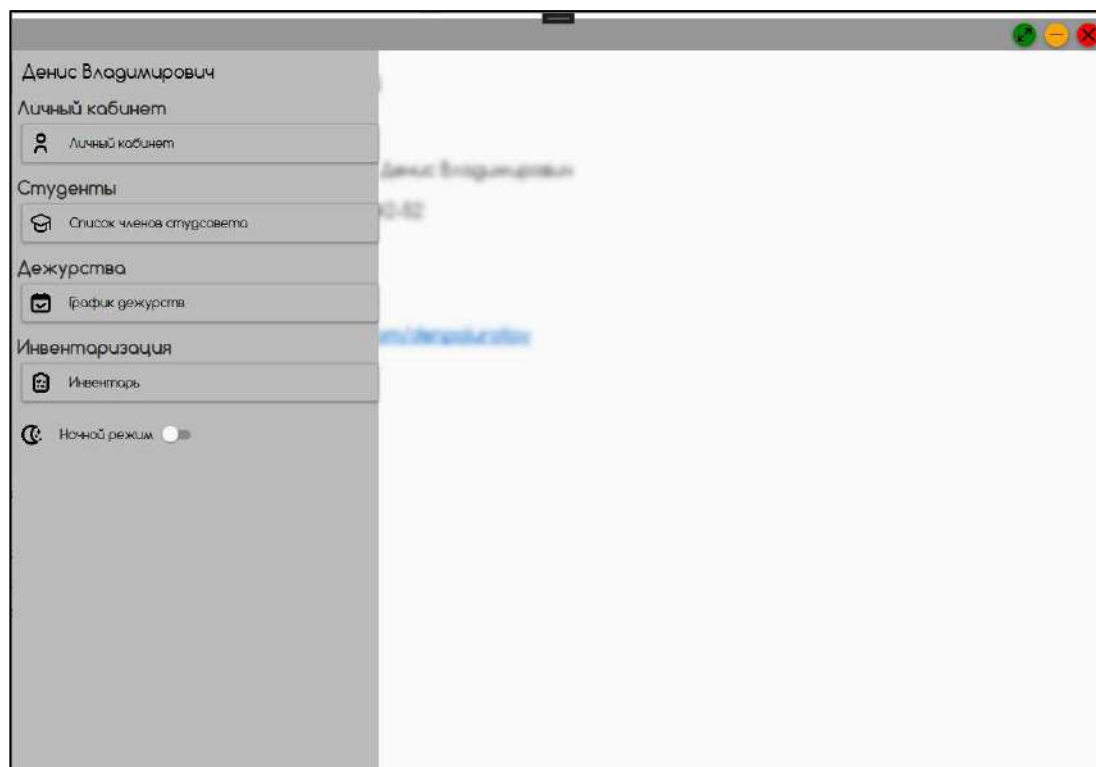


Рисунок 15 – Меню

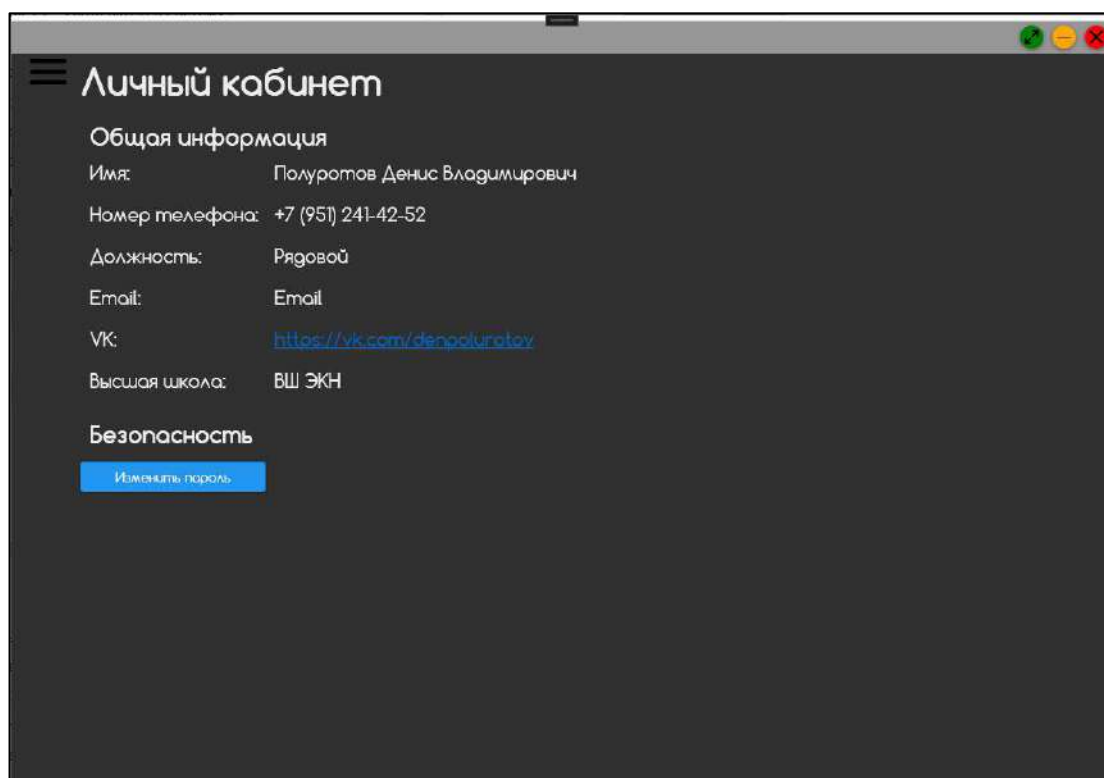


Рисунок 16 – Темная тема приложения

При нажатии кнопки «Список членов студсовета» пользователь попадет в соответствующий раздел, который представлен на рисунке 17. В данном разделе мы сразу видим список всех членов студенческого совета пользователя. При переключении флага «Архив» показываются и пропадают те участники студенческого совета, которые были исключены из студенческого совета, по той или иной причине. Для удобства и наглядности они подсвечиваются синим цветом.

Сверху раздел, под описанием, есть две кнопки – «Добавить нового студента» и «Обновить». При нажатии на кнопку «Обновить» система обновляет данные из базы данных. А при нажатии на кнопку «Добавить нового студента» открывается форма добавления нового члена студсовета, представленная на рисунке 18.

Члены студсовета

Список студентов, состоящих в студенческом совете

Кликните дважды по нужному студенту для получения подробной информации.

☒ Архив

ФИО	Номер группы	Дата рождения	Должность	Дата вступления	Архив
Ханназаров Ильяс Заирович	КЭ-111	12/04/2002	Участник	05/11/2022	<input type="checkbox"/>
Пашков Кирилл Евгеньевич	КЭ-116	15/11/2000	Участник	07/11/2022	<input type="checkbox"/>
Мицукова Ксения Станиславовна	КЭ-103	09/08/2002	Участник	06/11/2022	<input type="checkbox"/>
Синельникова Илона Андреевна	КЭ-101	22/05/2002	Участник	28/11/2022	<input type="checkbox"/>
Бутюгин Владислав Николаевич	КЭ-108	28/02/2002	Участник	06/11/2022	<input type="checkbox"/>
Сараева Анастасия Андреевна	КЭ-101	01/05/2002	Участник	06/11/2022	<input type="checkbox"/>
Понуратов Денис Владимирович	КЭ-403	06/10/1999	Участник	20/12/2019	<input type="checkbox"/>
Лебедев Никита Витальевич	КЭ-119	13/04/2003	Участник	28/11/2022	<input type="checkbox"/>
Абулгазина Анна Николаевна	КЭ-108	12/02/1999	Участник	04/12/2022	<input checked="" type="checkbox"/>
Обжолов Антон Игоревич	КЭ-115	04/02/2001	Председатель	20/12/2019	<input type="checkbox"/>

Рисунок 17 – Раздел «Список членов студсовета»

Члены студсовета

☒ Создание нового члена студсовета.

Выбор существующего студента: Студенты

Фамилия:

Имя:

Отчество:

Дата вступления:

Id профиля ВК:

Роль в студсовете:

Номер телефона:

Дата рождения:

Академическая группа:

Пол:

Зарегистрировать

Рисунок 18 – Форма создания нового члена студсовета

Если требуется добавить в студенческий совет человека, который ранее уже был добавлен в систему, то можно воспользоваться окном выбора существующего студента. Открывается оно путем нажатия на кнопку «Студенты». После этого все поля заполняются информацией, которая уже была внесена в базу данных. После заполнения всех полей, пользователь нажимает на кнопку «Зарегистрировать», после чего студент становится членом студенческого совета.

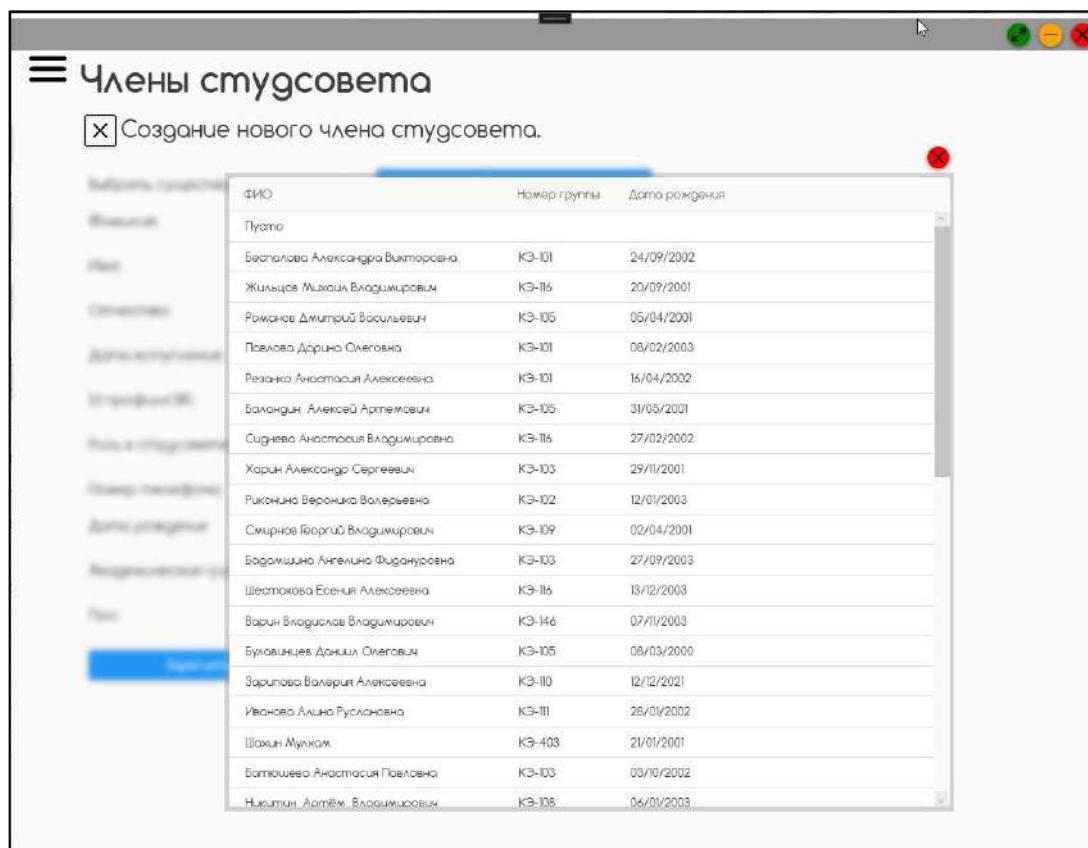
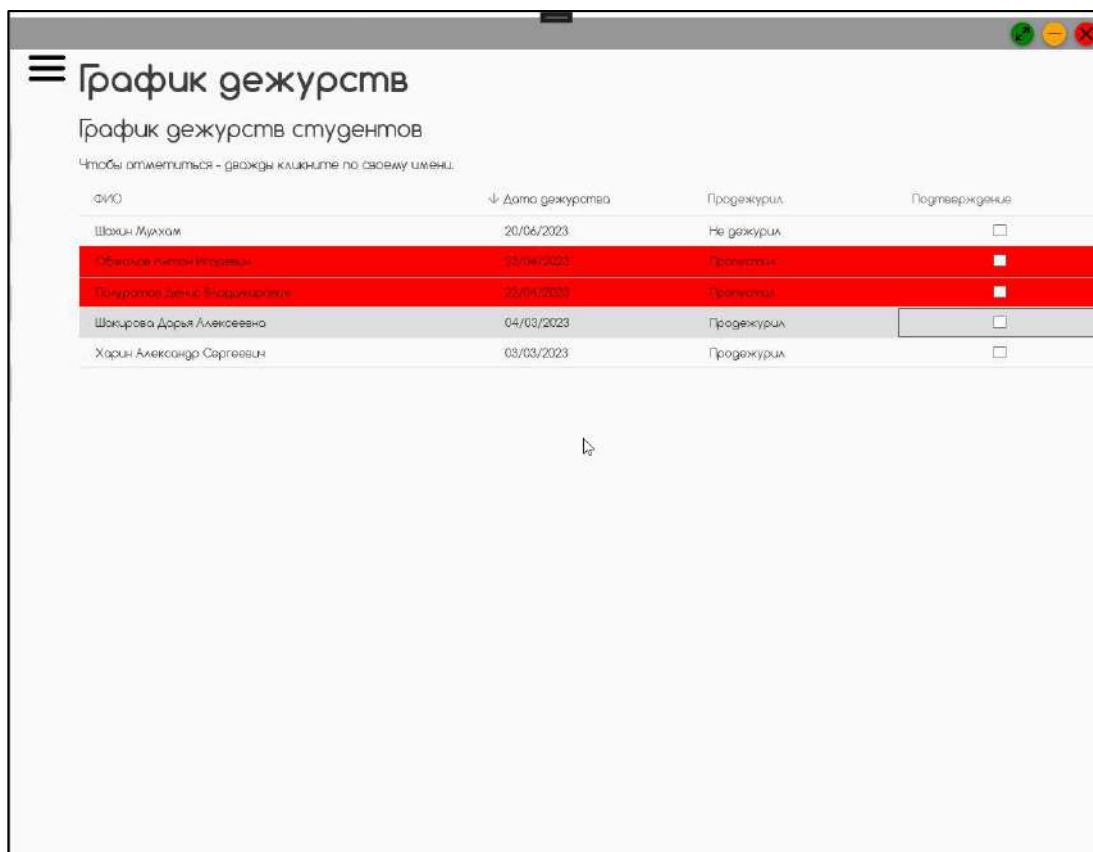


Рисунок 19 – Окно выбора студента

В разделе «График дежурств» пользователь видит список дежурств в его студенческом совете. Существует три состояния дежурства:

- 1) не дежурил;
- 2) продежурил;
- 3) пропустил.

Для того, чтобы отметить, пользователю необходимо кликнуть дважды по своему имени. Если пользователь пропустил свое дежурство, он или другие члены студенческого совета могут кликнуть правой кнопкой по его записи и выбрать пункт «Пропустил».



ФИО	Дата дежурства	Продежурил	Подтверждение
Шахин Мухом	20/06/2023	Не дежурил	<input type="checkbox"/>
Робакоев Аманжол	20/06/2023	Пропустил	<input checked="" type="checkbox"/>
Полудомов Денис Владимирович	20/06/2023	Пропустил	<input checked="" type="checkbox"/>
Шакирова Дарья Алексеевна	04/03/2023	Продежурил	<input type="checkbox"/>
Харин Александр Сергеевич	03/03/2023	Продежурил	<input type="checkbox"/>

Рисунок 20 – Раздел «График дежурств»

В разделе «Инвентарь» пользователь видит список актуального инвентаря своего студенческого совета. Имеется название единицы инвентаря, описание, и количество единиц. Для изменения единицы инвентаря необходимо дважды кликнуть по нужной записи и изменить данные в форме. Для создания новой записи необходимо нажать на кнопку над таблицей со знаком «+» и заполнить форму, после чего нажать на кнопку «Сохранить».

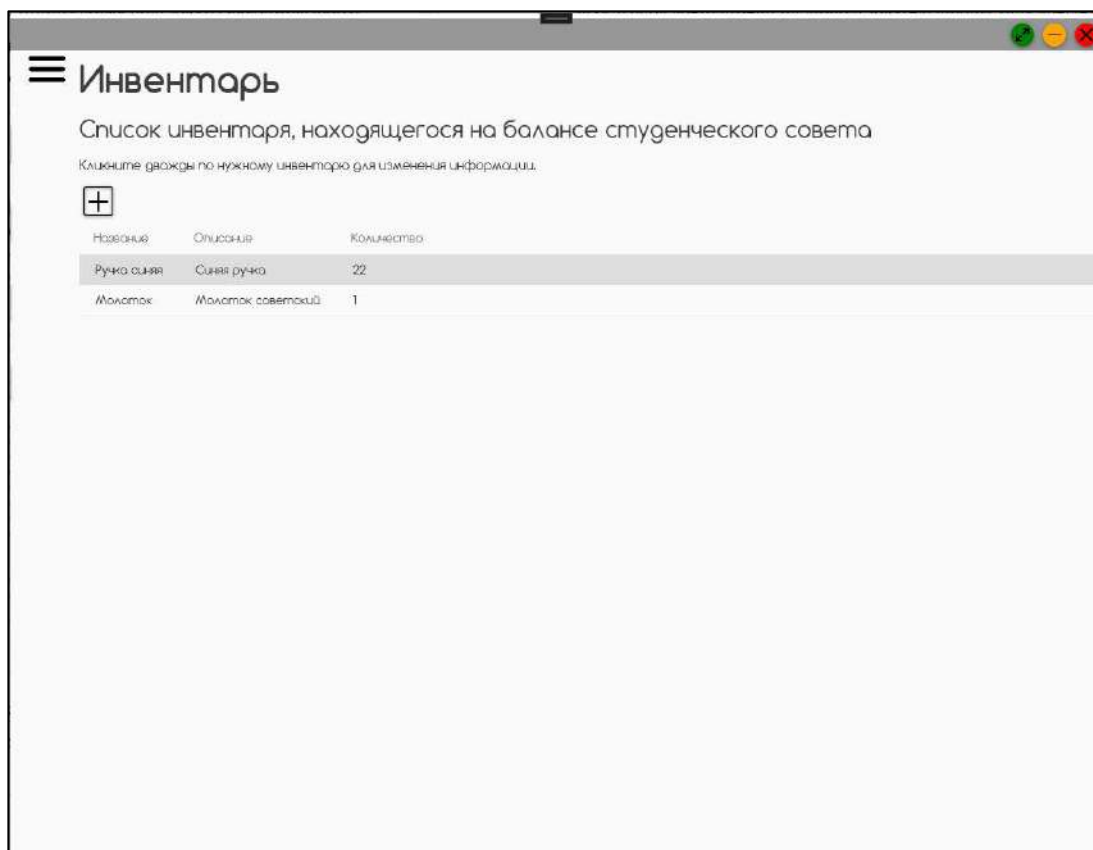


Рисунок 21 – Раздел «Инвентарь»

Вывод по четвертой главе

В результате реализации были созданы:

- 1) схема базы данных;
- 2) таблицы базы данных;
- 3) контроллеры в API;
- 4) Docker Container для работы API;
- 5) клиент.

После разработки необходимо провести функциональное тестирование системы.

5. ТЕСТИРОВАНИЕ СИСТЕМЫ.

5.1. Функциональное тестирование

Функциональное тестирование – это способ определить, работает ли программное обеспечение или приложение так, как ожидается. Функциональное тестирование интересуется не тем, как происходит обработка данных, а тем, обеспечивает ли она правильные результаты или имеет какие-либо ошибки [25]. Функциональные требования определяют, что именно делает ПО, какие задачи решает. Это необходимо для понимания, отвечает ли система заявленным требованиям и насколько она будет устраивать пользователя при использовании ее по назначению. Итоги функционального тестирования представлены в таблице 1.

Таблица 1 – Функциональное тестирование

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
1	Авторизация с корректными данными пользователя.	Пользователь успешно авторизован в системе и получен код 200 (OK).	При вводе корректных данных система авторизовала пользователя и вернула код 200 (OK)	Да
2	Авторизация с некорректными данными.	API вернет ошибку 401 (Unauthorized).	При вводе некорректных данных, был получен код 401 (Unauthorized).	Да
3	Получить состав студсовета авторизованного пользователя	API вернет код 200 (OK) и список членов студенческого совета пользователя в формате JSON.	API вернул код 200 (OK) и список членов студенческого совета пользователя в формате JSON.	Да
4	Получить список инвентаря студенческого совета пользователя	API вернет код 200 (OK) и список инвентаря студенческого совета пользователя в формате JSON.	API вернул код 200 (OK) и список инвентаря студенческого совета пользователя в формате JSON.	Да
5	Изменить пароль пользователя	API вернет код 200 (OK) и пароль для входа изменится.	API вернул код 200 (OK) и пароль был изменен.	Да

№	Название теста	Ожидаемый результат	Полученный результат	Тест пройден?
6	Изменить информацию об участнике студсовета	API вернет код 200 (ОК) и информация об участнике изменится.	API вернул код 200 (ОК) и информация об участнике изменилась.	Да
7	Изменить информацию об инвентаре студсовета	API вернет код 200 и информация об инвентаре изменится.	API вернул код 200 и информация об инвентаре изменилась.	Да
8	Создать нового члена студенческого совета	API вернет код 200 и создастся новый участник студсовета.	API вернул код 200 и создался новый участник студсовета.	Да
9	Создать новую запись единицы инвентаря	API вернет код 200 и появится новая запись о единице инвентаря.	API вернул код 200 и появилась новая запись о единице инвентаря.	Да
10	Изменить пароль пользователя	API вернет код 200 и пароль пользователя сменится.	API вернул код 200 и пароль пользователя сменился.	Да
11	Изменить данные пользователя	API вернет код 200 и данные пользователя изменятся.	API вернул код 200 и данные пользователя изменились.	Да

Вывод по пятой главе

В ходе разработки было проведено функциональное тестирование API. Тесты были пройдены успешно, были получены ожидаемые результаты, что указывает на то, что система работает так, как было задумано.

ЗАКЛЮЧЕНИЕ

В рамках данной работы был разработаны база данных и API для приложения студенческих советов ЮУрГУ. При этом были решены следующие задачи.

1. Создана база данных.
2. На сервере развернут Docker с базой данных и API.
3. Создан клиент.
4. Произведено тестирование API.

ЛИТЕРАТУРА

1. Что такое SHA-256. [Электронный ресурс] URL: <https://currency.com/ru/chto-takoe-sha-256> (дата обращения: 04.05.2023 г.).
2. Десктоп приложения. [Электронный ресурс] URL: <https://freematiq.com/uslugi/desktop-prilozheniya/> (дата обращения: 04.05.2023 г.).
3. Что такое фреймворк. Объясняем простыми словами. [Электронный ресурс] URL: <https://secretmag.ru/enciklopediya/chto-takoe-freimvork-obyasnyаем-prostymi-slovami.htm> (дата обращения: 04.05.2023 г.).
4. Что такое API? [Электронный ресурс] URL: <https://aws.amazon.com/ru/what-is/api/> (дата обращения: 20.02.2023 г.).
5. Что такое открытый исходный код? [Электронный ресурс] URL: <https://aws.amazon.com/ru/what-is/open-source/> (дата обращения: 04.05.2023 г.).
6. Что такое транзакция. [Электронный ресурс] URL: <https://habr.com/ru/articles/537594/> (дата обращения: 04.05.2023 г.).
7. Введение в ASP.NET Core. [Электронный ресурс] URL: <https://metanit.com/sharp/aspnet6/1.1.php> (дата обращения: 20.02.2023 г.).
8. REST API: что это простыми словами: принципы, стандарты, описание. [Электронный ресурс] URL: <https://boodet.online/blog/rest-api-chto-eto-prostymi-slovami-principy-standarty-opisanie> (дата обращения: 02.05.2023 г.).
9. Что такое реляционная база данных (РСУБД)? [Электронный ресурс] URL: <https://www.oracle.com/cis/database/what-is-a-relational-database> (дата обращения: 02.05.2023 г.).
10. Объектно-ориентированные базы данных. [Электронный ресурс] URL: <https://dic.academic.ru/dic.nsf/ruwiki/1071241> (дата обращения: 02.05.2023 г.).

11. Распределенные базы данных. [Электронный ресурс] URL: <http://www.kgau.ru/istiki/umk/ituman/textbox/bdras.htm> (дата обращения: 02.05.2023 г.).
12. Что такое хранилище данных? [Электронный ресурс] URL: <https://www.oracle.com/cis/database/what-is-a-data-warehouse/> (дата обращения: 02.05.2023 г.).
13. Что такое контейнеры? [Электронный ресурс] URL: <https://selectel.ru/blog/what-is-docker/> (дата обращения: 02.05.2023 г.).
14. Использование диаграммы вариантов использования UML при проектировании программного обеспечения. [Электронный ресурс] URL: <https://habr.com/ru/articles/566218/> (дата обращения: 02.05.2023 г.).
15. Новые возможности C# 11. [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/whats-new/csharp-11> (дата обращения: 02.05.2023).
16. Microsoft выпустила открытую платформу .NET 7. [Электронный ресурс] URL: <https://habr.com/ru/news/698736/> (дата обращения: 02.05.2023 г.).
17. Что такое схема базы данных? [Электронный ресурс] URL: <https://www.lucidchart.com/pages/ru/схемыбаз-данных> (дата обращения: 04.05.2023 г.).
18. Создание таблиц (ядро СУБД). [Электронный ресурс] URL: <https://learn.microsoft.com/ru-ru/sql/relational-databases/tables/create-tables-database-engine?view=sql-server-ver16> (дата обращения: 04.05.2023 г.).
19. Как создать простое Rest API на .NET Core. [Электронный ресурс] URL: <https://habr.com/ru/articles/531106/> (дата обращения: 04.05.2023 г.).

20. Контроллеры. [Электронный ресурс] URL:
<https://metanit.com/sharp/aspnetmvc/2.1.php#:~:text=B%20ASP.NET%20Core%20MVC,иметь%20поля%2C%20свойства%2C%20методы.> (дата обращения: 04.05.2023 г.).
21. Принципы построения REST JSON API. [Электронный ресурс] URL: <https://habr.com/ru/post/447322/> (дата обращения: 20.02.2023 г.).
22. Пакуем приложения ASP.NET Core с помощью Docker. [Электронный ресурс] URL:
<https://habr.com/ru/companies/microsoft/articles/435914/> (дата обращения: 04.05.2023 г.).
23. Try Docker Compose. [Электронный ресурс] URL:
<https://docs.docker.com/compose/gettingstarted/> (дата обращения: 04.05.2023 г.).
24. Docker Compose – Docker: Основы. [Электронный ресурс] URL:
https://ru.hexlet.io/courses/docker-basics/lessons/docker-compose/theory_unit (дата обращения: 04.05.2023 г.).
25. Что такое функциональное тестирование? Типы, примеры, контрольный список и реализация. [Электронный ресурс] URL:
<https://www.zaptest.com/ru/что-такое-функциональное-тестирован> (дата обращения: 04.05.2023 г.).

ПРИЛОЖЕНИЕ. Спецификация вариантов использования

Спецификация вариантов использования (ВИ) системы приведена в таблицах 1–7.

Таблица 1 – Спецификация ВИ «Посмотреть состав студсовета»

Прецедент: Посмотреть состав студенческого совета
ID: 1
Аннотация: Просмотреть действующий и архивный состав своего студенческого совета
Главные актеры: Участник студсовета
Второстепенные актеры: Зам. председателя студсовета, Председатель студсовета
Предусловия: 1. Пользователь авторизован
Основной поток: 1. Вариант использования начинается тогда, когда Пользователь отправляет запрос на получение состава студсовета. 2. Пользователь просматривает список студентов.
Постусловия: 1. Пользователь посмотрел список студентов.
Альтернативные потоки: 1. Пользователь просматривает архивный состав студсовета.

Таблица 2 – Спецификация ВИ «Посмотреть инвентарь студсовета»

Прецедент: Посмотреть инвентарь студенческого совета
ID: 2
Аннотация: Просмотреть инвентарь своего студенческого совета
Главные актеры: Участник студсовета
Второстепенные актеры: Зам. председателя студсовета, Председатель студсовета
Предусловия: 1. Пользователь авторизован.
Основной поток: 1. Вариант использования начинается тогда, когда Пользователь отправляет запрос на получение списка инвентаря. 2. Пользователь просматривает список инвентаря.
Постусловия: 1. Пользователь посмотрел список инвентаря
Альтернативные потоки: Нет

Таблица 3 – Спецификация ВИ «Посмотреть состав студсовета»

Прецедент: Посмотреть состав студенческого совета
ID: 3
Аннотация: Просмотреть действующий и архивный состав своего студенческого совета
Главные актеры: Участник студсовета
Второстепенные актеры: Зам. председателя студсовета, Председатель студсовета
Предусловия: 1. Пользователь авторизован
Основной поток: 1. Вариант использования начинается тогда, когда Пользователь отправляет запрос на получение состава студсовета. 2. Пользователь просматривает список студентов.
Постусловия: 1. Пользователь посмотрел список студентов.
Альтернативные потоки: 1. Пользователь просматривает архивный состав студсовета.

Таблица 4 – Спецификация ВИ «Исправить информацию о человеке из студсовета»

Прецедент: Исправить информацию о человеке из студсовета
ID: 4
Аннотация: Исправить информацию о человеке из своего студенческого совета
Главные актеры: Зам. председателя студсовета
Второстепенные актеры: Председатель студсовета
Предусловия: 1. Пользователь авторизован как председатель или зам. председателя
Основной поток: 1. Вариант использования начинается тогда, когда Пользователь выбирает участника студсовета. 2. Пользователь изменяет данные человека 3. Пользователь отправляет запрос на изменение информации о человеке при помощи строки JSON. 4. Пользователь получает ответ от API.
Постусловия: 1. Система сохраняет изменения
Альтернативные потоки: 1. Пользователь неверно заполнил данные человека, и система вернула ему код 400.

Таблица 5 – Спецификация ВИ «Исправить информацию об инвентаре»

Прецедент: Исправить информацию об инвентаре
ID: 5
Аннотация: Исправить информацию об инвентаре
Главные актеры: Зам. председателя студсовета
Второстепенные актеры: Председатель студсовета
Предусловия: 1. Пользователь авторизован как председатель или зам. председателя
Основной поток: 1. Вариант использования начинается тогда, когда Пользователь выбирает единицу инвентаря студсовета. 2. Пользователь изменяет данные единицы. 3. Пользователь отправляет запрос на изменение информации о единице инвентаря при помощи строки JSON. 4. Пользователь получает ответ от API.
Постусловия: 1. Система сохраняет изменения.
Альтернативные потоки: 1. Пользователь неверно заполнил данные и система вернула ему код 400.

Таблица 6 – Спецификация ВИ «Создать нового члена студенческого совета»

Прецедент: Создать нового члена студенческого совета
ID: 6
Аннотация: Создать нового члена студенческого совета
Главные актеры: Председатель студсовета
Второстепенные актеры: Нет
Предусловия: 1. Пользователь авторизован как председатель
Основной поток: 1. Вариант использования начинается, когда Пользователь начинает вводить данные нового члена студенческого совета. 2. Пользователь отправляет запрос на создание информации о человеке при помощи строки JSON. 3. Пользователь получает ответ от API.
Постусловия: 1. Система сохраняет изменения
Альтернативные потоки: 1. Пользователь неверно заполнил данные человека, и система вернула ему код 400.

Таблица 7 – Спецификация ВИ «Изменить пароль пользователя»

Прецедент: Изменить пароль пользователя
ID: 7
Аннотация: Изменить пароль пользователя
Главные актеры: Администратор
Второстепенные актеры: Нет
Предусловия: 1. Пользователь авторизован как администратор
Основной поток: 1. Вариант использования начинается, когда Администратор выбирает пользователя. 2. Администратор отправляет запрос на создание информации о человеке при помощи строки JSON. 3. Пользователь получает ответ от API.
Постусловия: 1. Система сохраняет изменения
Альтернативные потоки: 1. Пользователь неверно заполнил данные человека, и система вернула ему код 400.