МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Компьютерный клуб»

Исполнитель

студент (ка) 2 курса группы 6 Быков П.А.

(Ф.И.О.)

Руководитель работы ст. препод. Сухорукова И.Г.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Мущук А.Н.

(подпись)

Минск 2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc167101278)

[1.1. Анализ прототипов 4](#_Toc167101279)

[1.2. Требования к проекту 8](#_Toc167101280)

[2. Анализ требований к программному средству и разработка функциональных требований 9](#_Toc167101281)

[2.1. Описание средств разработки 9](#_Toc167101282)

[2.2. Спецификация функциональных требований к программному средству 11](#_Toc167101283)

[2.3. Спецификация функциональных требований 12](#_Toc167101284)

[3. Проектирование программного средства 13](#_Toc167101285)

[3.1. Общая структура 13](#_Toc167101286)

[3.2. Взаимоотношения между классами 14](#_Toc167101287)

[3.3. Модель базы данных 14](#_Toc167101288)

[3.5. Проектирование последовательностей проекта 16](#_Toc167101289)

[4.1. Основные классы программного средства 17](#_Toc167101290)

[4.2. Описание классов и методов программного средства 17](#_Toc167101291)

[4.2.1. Выполнение входа 17](#_Toc167101292)

[4.2.2. Добавление пользователей 17](#_Toc167101293)

[4.2.3. Управление тарифами 18](#_Toc167101294)

[4.2.4. Управление компьютерами 18](#_Toc167101295)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 20](#_Toc167101296)

[5.1. Тестирование авторизации 20](#_Toc167101297)

[5.2. Тестирование создания тарифа 21](#_Toc167101298)

[5.3. Тестирование удаления тарифа 22](#_Toc167101299)

[5.4. Тестирование добавления сотрудника 23](#_Toc167101300)

[Выводы по разделу: 24](#_Toc167101301)

[6. Руководство по установке и использованию 25](#_Toc167101302)

[Выводы по разделу: 30](#_Toc167101303)

[Заключение 31](#_Toc167101304)

[Список литературы 32](#_Toc167101305)

[ПРИЛОЖЕНИЕ А 33](#_Toc167101306)

[ПРИЛОЖЕНИЕ Б 34](#_Toc167101307)

[ПРИЛОЖЕНИЕ В 35](#_Toc167101308)

[ПРИЛОЖЕНИЕ Г 36](#_Toc167101309)

## ВВЕДЕНИЕ

В современном мире компьютерные клубы являются популярным местом среди любителей видеоигр, профессиональных геймеров, студентов и профессионалов в области информационных технологий. Управление такими заведениями требует эффективной системы контроля доступа, учета времени использования оборудования, а также мониторинга работы компьютеров и активности пользователей.

В рамках данного курсового проекта предлагается разработка программного средства для управления компьютерным клубом с использованием языка программирования C# и технологии Windows Presentation Foundation (WPF). Данное программное обеспечение будет предоставлять администратору клуба инструменты для эффективного контроля и управления всеми аспектами его работы.

Целью данного проекта является создание интуитивно понятного и функционального приложения, способного упростить рутинные задачи администрирования компьютерного клуба, повысить эффективность работы персонала и обеспечить удобство для посетителей.

В ходе разработки будут рассмотрены основные аспекты проектирования и реализации программного средства, включая архитектуру приложения, пользовательский интерфейс, механизмы управления доступом, а также интеграцию с базой данных для хранения информации о пользователях, времени использования и других важных параметров.

Кроме того, будут рассмотрены возможности расширения функционала программного средства, включая интеграцию с системами онлайн-платежей, создание отчетности о работе клуба и многое другое.

В результате успешной реализации данного проекта ожидается создание удобного и мощного инструмента для управления компьютерным клубом, способного повысить качество обслуживания клиентов и оптимизировать рабочие процессы администрации

**1**.**Аналитический обзор литературы и формирование требований**

## Анализ прототипов

Были проанализированы цели и задачи, поставленные в данном курсовом проекте, а также рассмотрены аналогичные примеры их решений. На основании анализа всех достоинств и недостатков данных альтернативных решений были сформулированы требования к данному программному средству.

* + 1. **Smartshell**

Современное программное обеспечение для управления компьютерным клубом. Программный комплекс включает всё необходимое для эффективного управления клубом: удобный дашборд с виджетами, магазин и склад, мониторинг оборудования, отчёты по сменам, подробные логи событий, клубные аккаунты, акции, скидки, промокоды, кассы, уведомления в Telegram и другое. Интерфейс представлен на рисунке 1.1.

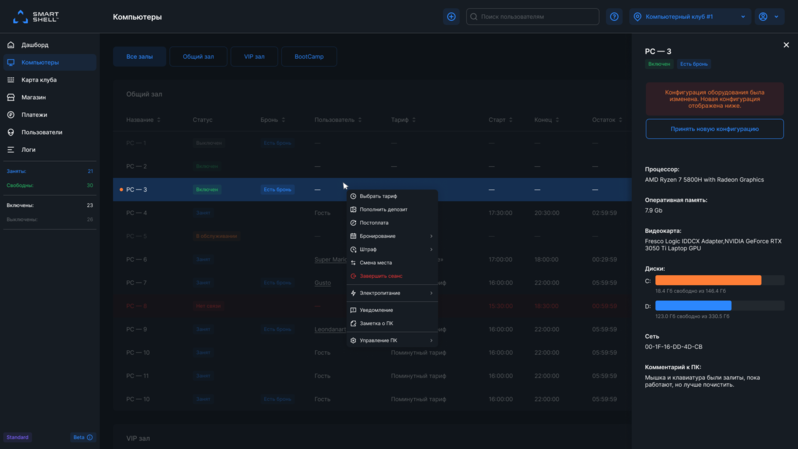


Рисунок 1.1 – Интерфейс Smartshell

Явным недостатком является избыточность информации. Например, информация о комплектующих каждого отдельного компьютера не может быть полезна пользователю, и лишь усложняет взаимодействие с программой.

* + 1. AstrumSoft

Программа "AstrumSoft" предназначена для автоматизации работы компьютерных клубов, интернет кафе. Учет оплаченного времени использования компьютеров, игровых приставок или любых других устройств, продажи дополнительных услуг (кофе, бутерброды). Компьютеры за которые не заплачено или вышло время блокируются. Увеличивает доход от работы клуба, следя за администраторами (предотвращает воровство).

Серьёзным недостатком данного программного обеспечения является устаревший, визуально перегруженный интерфейс. Кроме того доступ к функционалу осуществляется интуитивно непонятными способами. Это видно на рисунке 1.2.

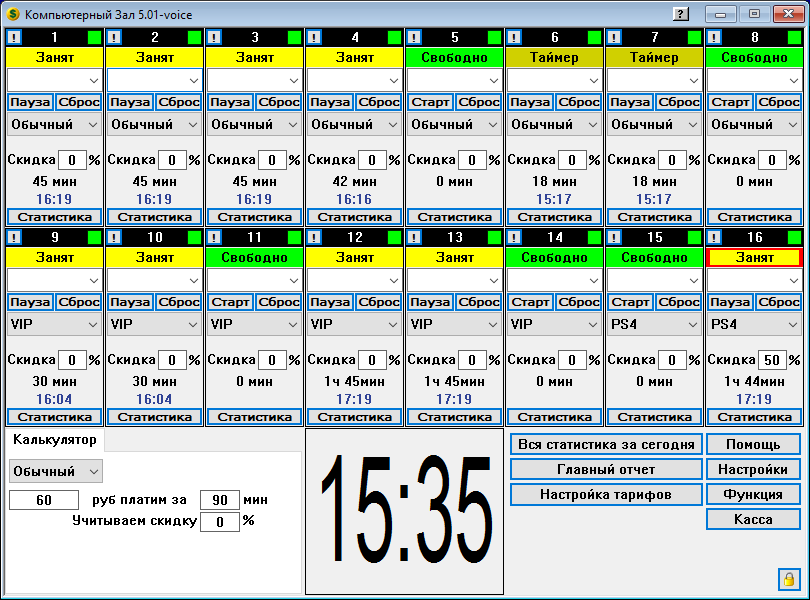


Рисунок 1.2 – Интерфейс AstrumSoft

Кроме того, программное средство обеспечивает подробные отчеты, защищенные от изменений, полную поддержку Windows 11. Компьютеры клиентов находятся под надежным контролем.

* + 1. Lightshell

LightShell представляет собой инновационное программное обеспечение, разработанное с учетом потребностей компьютерных клубов. Интерфейс Lightshell использует крайне вычурные цвета, что не является его плюсом. Это продемонстрированно на рисунке 1.3.

В основе LightShell лежит идея управления категориями клиентов. Это позволяет вам легко классифицировать ваших клиентов и предлагать им наиболее подходящие услуги. LightShell также предлагает уникальные возможности для управления пакетами услуг. Вы можете настраивать как сгораемые, так и не сгораемые пакеты услуг, чтобы удовлетворить потребности каждого клиента.

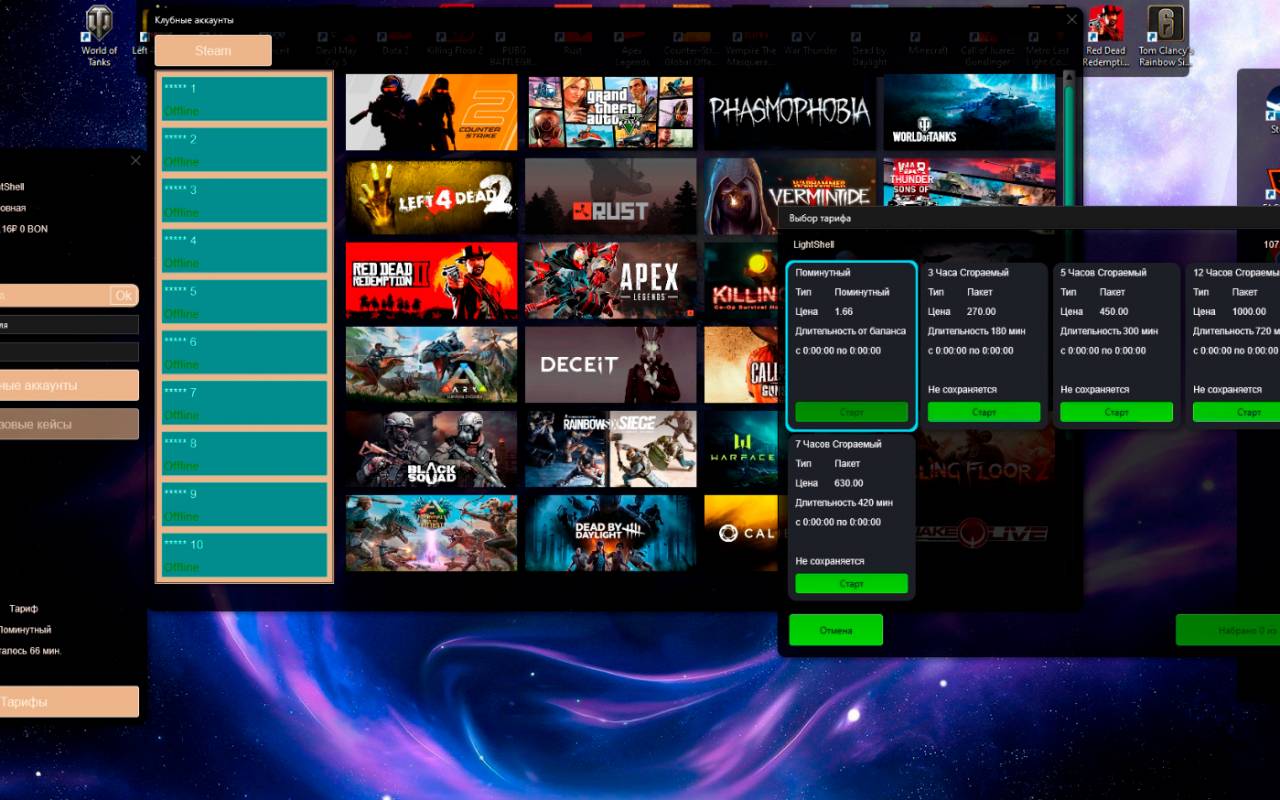


Рисунок 1.3 – Интерфейс Lightshell

Основным недостатком LightShell является отсутствие гибкого контроля над каждым компьютером в зале.

* + 1. ggLeap

Управляйте своим игровым центром из любого места. Наше облачное решение позволяет вам иметь полный контроль на клубом, где бы вы ни находились, даже на планшете или телефоне. Упрощение ведения вашего бизнеса. ggLeap использует передовые технологии для обеспечения наилучшего возможного опыта. Помимо наших собственных решений, ggLeap имеет интеграцию с высококачественными сторонними решениями, такими как Square. Интерфейс программы представлен на рисунке 1.5.



Рисунок 1.4 – Интерфейс ggLeap

Данная система является слишком нагруженной, а потому она не очень хорошо подходит для небольших компьютерных клубов.

* + 1. Smartlaunch

Smartlaunch — передовое программное обеспечение для управления спортивными и игровыми центрами. Smartlaunch включает в себя полный функционал POS для продажи товаров и услуг, а также отслеживание времени практически всего: от использования компьютера до аренды конференц-залов. Средство подходит для мониторинга и управления высококлассными компьютерами, консолями, виртуальной реальностью и большими дисплеями для потоковой передачи, для размещения и взимания платы за турниры и мероприятия, для продажи билетов с предоплатой, создания программ членства и лояльности и многого-многого другого.

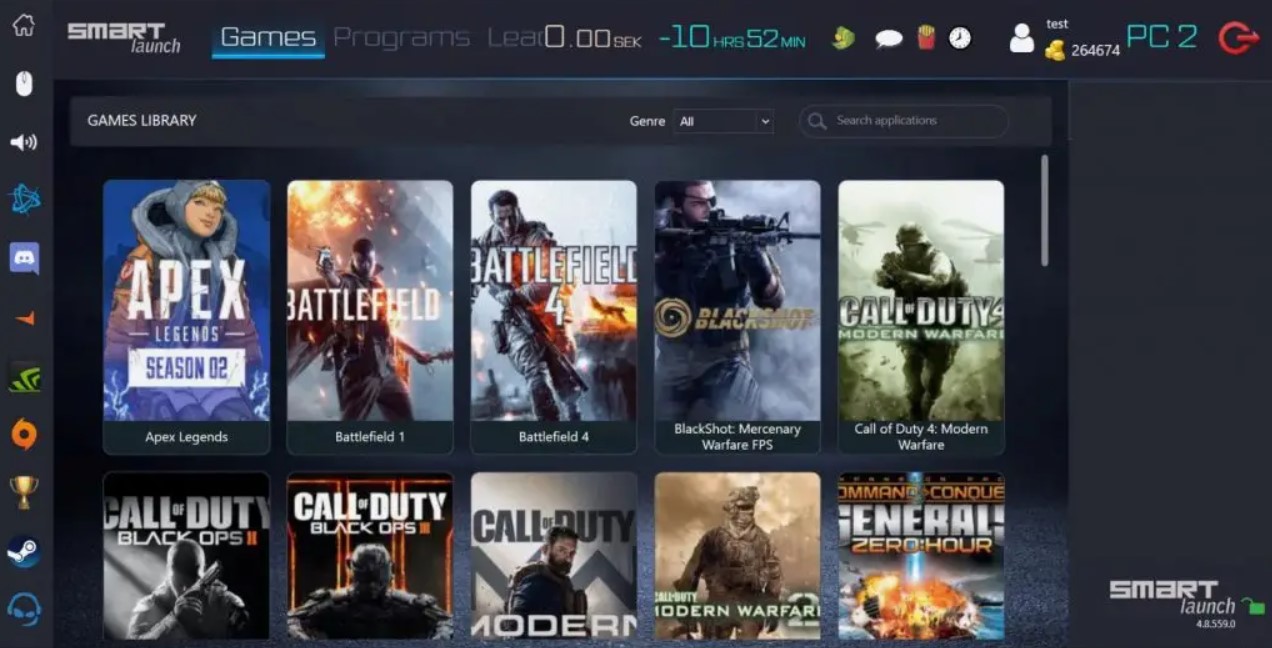


Рисунок 1.5 – Интерфейс Smartlauch

Главным минусом данного программного средства является неразрывная связь двух частей приложения: приемника и менеджера.

## Требования к проекту

Обзор вышеперечисленных известных аналогов позволяет проанализировать все преимущества и недостатки альтернативных возможностей и позволяет сформулировать список требований, предъявляемых к программному средству, разрабатываемому в данном курсовом проекте. Программное средство должно обеспечивать возможность выполнения перечисленных ниже функций:

* поддержка множества компьютерных клубов одновременно;
* возможность заранее создавать тарифы для ускорения обслуживания;
* возможность регулировать состояния каждого компьютера;
* возможность следить за историей операций;
* ведение подсчета доходов основываясь на тарифах и времени.

**Выводы по разделу:**

В разделе были определены ключевой функционал для программного средства, самыми важными из которых являются:

* возможность заранее создавать тарифы для ускорения обслуживания;
* возможность регулировать состояния каждого компьютера.

Так же программное средство должно целевую платформу Windows 10 и иметь дизайн в темных тонах.

## 2. Анализ требований к программному средству и разработка функциональных требований

# **Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET Core 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* технология Fluent API;
* база данных MS SQL Server.

1. **Microsoft Visual Studio 2022**

Microsoft Visual Studio 2022 — это интегрированная среда разработки для написания, отладки и сборки кода, а также последующей публикации приложений. Данный продукт позволяет разрабатывать не только консольные, но и десктопные приложения, с использованиям таких технологий, как WinForms или WPF.

1. **Программная платформа .NET Core 8.0**

Платформа .NET Core — это созданная Microsoft в 2016 году кроссплатформенная, с открытым исходным кодом технология, предназначенная для создания и выполнения современных приложений. Она поддерживает разработку различных типов приложений, включая веб-приложения, облачные сервисы, десктопные приложения и микросервисы, работая на операционных системах Windows, macOS и Linux.

Основой .NET Core является общеязыковая среда выполнения Common Language Runtime (CLR), которая обеспечивает выполнение кода, написанного на различных поддерживаемых языках программирования, таких как C#, F#, и Visual Basic. .NET Core предлагает модульную архитектуру, где разработчики могут выбирать и использовать только необходимые пакеты, что позволяет создавать более легкие и производительные приложения.

1. **Язык программирования C#**

В качестве языка программирования используется C# – основной язык разработки в .NET Framework. Язык объектно-ориентированный, имеет строгую статическую типизацию, поддерживает перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения. Используется как основной язык в технологии WPF.

1. **Технология WPF**

Для предоставления пользовательского интерфейса и разграничения дизайна и бизнес-логики используется технология Microsoft WPF – аналог WinForms, система для построения клиентских приложений Windows с возможностями взаимодействия с пользователем и графическая подсистема в составе .NET Framework, использующая язык разметки XAML.

1. **Расширяемый язык разметки XAML**

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. XAML представляет собой язык декларативного описания интерфейса, основанный на XML.

1. **Технология Entity Framework Core**

Для взаимодействия с базой данных в приложении WPF используется технология Entity Framework Core — кроссплатформенная и расширяемая версия технологии Entity Framework с открытым исходным кодом, которая активно развивается и дополняется. В то время как традиционные средства ADO.NET обеспечивают создание подключений, выполнение команд и работу с другими объектами для взаимодействия с базами данных, Entity Framework предлагает более высокий уровень абстракции. Эта технология позволяет разработчикам абстрагироваться от конкретной базы данных и работать с данными независимо от типа хранилища. Entity Framework Core является последней версией Entity Framework 7, которая совместима с .NET Core и продолжает активно развиваться.  
 Entity Framework Core предоставляет три основных способа взаимодействия с базой данных:

* Database First: Entity Framework Core создает набор классов, отражающих модель существующей базы данных.
* Model First: разработчик сначала создает модель данных, на основе которой Entity Framework Core затем генерирует реальную базу данных на сервере.
* Code First: разработчик создает классы моделей данных, которые будут храниться в базе данных, а затем Entity Framework Core генерирует базу данных и ее таблицы на основе этих моделей.

1. **Технология Fluent API**

При настройке отображения модели базы данных в Entity Framework Core могут использоваться два подхода: анотации и Fluent API. В данном проекте для реализации поставленных технических задач был выбран последний подход. Технология Entity Framework Core также способна самостоятельно сопоставить классы моделей с таблицами базы данных при использования согласовании об именовании, однако Fluent API является набором методов, предоставляющим их более тонкую настройку императивным методом.

1. **MS SQL Server**

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта. Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос 10 на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения. Основной используемый язык запросов — Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

# **Спецификация функциональных требований к программному средству**

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* авторизация;
* просмотр списка компьютеров;
* управление состоянием компьютера (аренды);
* просмотр состояния компьютера (аренды);
* добавление новых компьютеров;
* просмотр списка тарифов;
* управление существующими тарифами;
* добавление новых тарифов;
* просмотр содержимого логов;
* экспорт логов в файл;
* получение статистики доходов за определенный срок;
* возможность начать смену;
* возможность сменить выбранный клуб из списка доступных;
* возможность валюту в приложении.

Для администратора:

* все функции пользователя;
* добавление новых сотрудников;
* редактирование информации о сотрудниках;
* удаление сотрудников;
* добавление новых клубов;
* удаление существующих клубов.

# **Спецификация функциональных требований**

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно база данных описано в следующем разделе.

В программном средстве необходимо реализовать авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Введенные данные, успешно прошедшие валидацию, сверяются с данными в базе данных.

Пользователь может управлять тарифами, компьютерами, арендами. Все сведения должны автоматически обновляться в базе данных.

Администратор обладает теми же возможностями, что и пользователь, а также обладает возможностями добавления, просмотра, редактирования и удаления информации о сотрудниках и добавления, удаления клубов.

**Выводы по разделу:**

В разделе были определены инструменты которые будут использоваться в программном средстве. К таким относятся:

* программная платформа .NET Core 8.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* технология Fluent API;
* база данных MS SQL Server.

## 3. Проектирование программного средства

# **Общая структура**

Программное средство использует архитектуру MVVM, а также архитектуру FSD (на слое View). Программное средство имеет следующую структуру, представленную на рисунке 3.1.

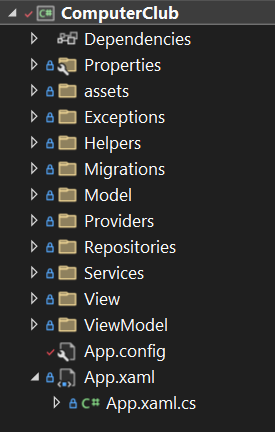


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| App.config | Файл с параметрами проекта. |
| App.xaml | Файл с общими ресурсами проекта |
| Папка assets | Папка, содержащая все изображения, шрифты и словари ресурсов. |
| Папка Exceptions | Папка, содержащая классы исключений. |
| Папка Helpers | Папка, содержащая классы, необходимые в работе других классов. |
| Папка Migrations | Папка, содержащая классы миграций базы данных Entity Framework Core. |
| Папка Model | Папка, содержащая классы, которые являются сущностями базы данных. |
| Папка Providers | Папка, содержащая классы-провайдеры. |
| Папка Repositories | Папка, содержащая классы, реализующие паттерн репозиторий. |
| Папка Services | Папка, содержащая классы-сервисы. |
| Папка View | Папка, содержащая файлы разметок и классы соответствующих страниц. |
| Папка ViewModel | Папка, содержащая классы являющимися контекстами для страниц. |

# **Взаимоотношения между классами**

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в Приложении А.

# **Модель базы данных**

Для реализации поставленной задачи была создана база данных ComputerClub. Для ее создания использовалась система управления реляционными базами данных MSQL Server. База данных состоит из таблиц, представленных на рисунке 3.3.

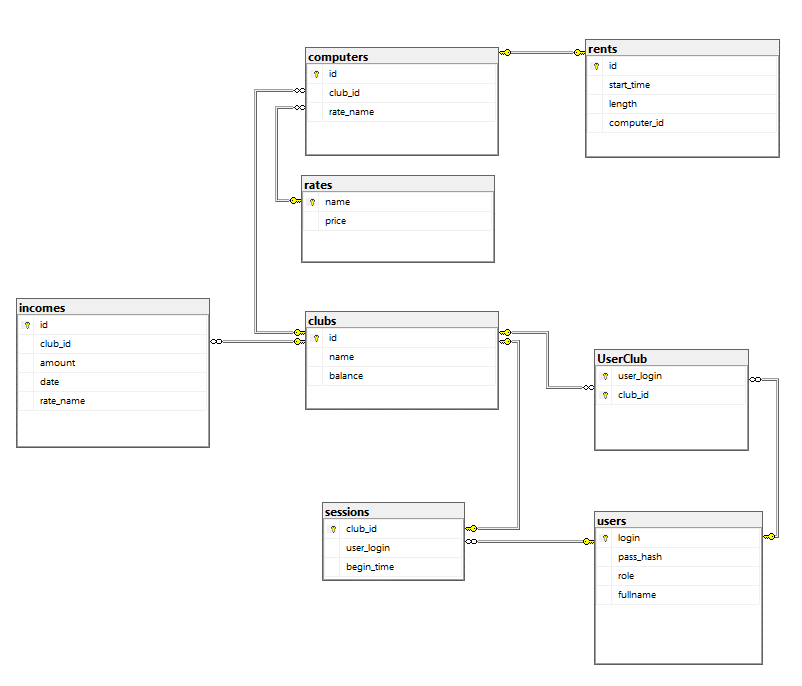


Рисунок 3.3 – База данных ComputerClub

В базе данных находится 8 таблиц: computers, rents, rates, clubs, incomes, sessions, users и UserClub.  При этом таблица UserClub является сводной таблицей, между таблицами users и clubs. Таблицы связаны между собой следующим образом:

* Таблица rents содержит информацию о времени аренды, длительности и компьютерах.
* Таблица rates определяет тарифы и их стоимость.
* Таблица clubs представляет собой клубы с балансом.
* Таблица incomes содержит данные о доходах клубов по тарифам.
* Таблица sessions отслеживает сессии пользователей в клубах.
* Таблица users хранит данные пользователей, включая логины, пароли, роли и полные имена.
* Таблица UserClub связывает пользователей с клубами, к которым они принадлежат.

**3.4. Проектирование архитектуры проекта**

Для общего представления функционального назначения системы используется диаграмма использования, описывающая, какой функционал разрабатываемой программной системы доступен каждой группе пользователей. На диаграмме использования применяются два типа основных сущностей: варианты использования и группы пользователей.

Разные группы пользователей в диаграмме называются актёрами, и обозначают любые сущности, использующие систему. Любая функция системы называется вариантом использования. Каждый вариант использования обозначает набор действий, который может быть использован актёром для взаимодействии с системой, и определяет набор действий, выполняемых этой системой.

Диаграмма использования представлена в приложении Б.

# **3.5. Проектирование последовательностей проекта**

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML-диаграмма – диаграмма последовательностей. Данная диаграмма иллюстрирует, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Для отображения течения времени используется линия жизни объекта, которая изображается с помощью штриховой линии, которая проводится вертикально вниз. С помощью линии жизни показывается период, в течение которого объект существует в системе. Сами объекты изображаются в виде прямоугольников, а сообщения, которыми они обмениваются – в виде линий со стрелками. Диаграмма последовательностей представлена в приложении В.

**Выводы по разделу:**

В разделе была выбрана архитектура проекта, которая задаёт общий план проектирование продукта. Также для программного средства были спроектированы сущности базы данных, на основе которых получены их аналоги в языке программирования C#.

**4. Реализация программного средства**

# **Основные классы программного средства**

Для выполнения технических задач программного средства должны быть реализованы следующие функции и соответствующие им классы и методы:

* Выполнение входа;
* Добавление пользователей;
* Управление компьютерами;
* Управление тарифами;
* Управление арендами.

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы и их функционал и реализация.

# **Описание классов и методов программного средства**

## Выполнение входа

Выполнение входа осуществляется в отдельном окне – LoginWindow. В своём контексте данное окно ссылается на сервис авторизации AuthService. Данный сервис характеризуется как:

* AuthService реализован как Singleton, что позволяет использовать только один экземпляр этого класса;
* Метод GetInstance возвращает единственный экземпляр класса AuthService;
* Метод TryAuth проверяет логин и хэш пароля пользователя. Если данные совпадают, он устанавливает текущего пользователя и клуб;
* Метод GetHash используется для получения MD5-хэша пароля;
* В конструкторе загружаются данные из таблиц Sessions, Clubs и Users;
* Метод GetAvailableClubs возвращает список клубов, к которым принадлежит текущий пользователь;
* Метод Clear сбрасывает текущего пользователя и клуб, очищая состояние сервиса.

Код класса AuthService представлен в приложении Г.

## Добавление пользователей

Для добавления новых пользователей используется страница сотрудники, доступная пользователям роли owner. Для этого необходимо заполнить форму, заполнить такие поля как: ФИО, логин, пароль, выбрать необходимые клубы. При успешной проверке валидации, страница обращается к UsersRepository и создает новый экземпляр класса User с полученными ранее параметрами.

## Управление тарифами

После успешного выполнения обновления списка тарифов пользователь взаимодействует с главной страницей тарификации, представленной классом RatesPageVM. Эта страница позволяет просматривать и управлять тарифами.

В конструкторе RatesPageVM инициализируется репозиторий тарифов и вызывается метод UpdateRates(), загружающий все текущие тарифы.

Метод AddRate открывает модальное окно AddRateModalWindow для добавления нового тарифа. После закрытия окна список тарифов обновляется. Метод RemoveRate удаляет выбранный тариф, проверяя наличие связанных компьютеров. Если такие компьютеры есть, выбрасывается исключение и показывается сообщение об ошибке. В противном случае тариф удаляется из репозитория и список обновляется.

Метод EditRate открывает модальное окно EditRateModalWindow для редактирования выбранного тарифа. После внесения изменений список тарифов обновляется. Метод UpdateRates загружает тарифы из репозитория и обновляет список.

Для добавления и редактирования тарифов используются модальные окна, позволяющие вводить и изменять данные. Исключения, возникающие при удалении тарифа, обрабатываются и отображаются пользователю через модальные окна уведомлений.

## Управление компьютерами

После успешного обновления списка компьютеров пользователь взаимодействует с главной страницей управления компьютерами, представленной классом ComputersPageVM.

Эта страница позволяет просматривать и управлять компьютерами. В конструкторе ComputersPageVM инициализируется репозиторий компьютеров и таймер аренды. Таймер обновляет отображение списка компьютеров через определенные промежутки времени. Также добавляются обработчики изменений свойств службы аутентификации и изменений в базе данных, которые вызывают обновление данных компьютеров. Метод ShowAddNewComputerWindow открывает модальное окно для добавления нового компьютера. Метод OpenComputerDetails открывает модальное окно с подробной информацией о выбранном компьютере.

Если возникает ошибка, пользователю показывается сообщение. Обработчик AuthServiceChangesHandler обновляет данные компьютеров при изменении текущего клуба в службе аутентификации. Обработчик DatabaseChangesHandler обновляет данные при изменениях в базе данных. Метод UpdateComputersData загружает и обновляет список компьютеров из репозитория.

**4.4.5. Управление арендами**

Управление арендами осуществляется на страницах каждого, отдельно взятого компьютера, при нажатии на кнопку изменить, поля становятся редактируемыми и при подтверждении изменений создается или редактируется существующая аренда с полученными данными.

Функционал "Управление арендами" в классе DetailedComputerWindowVM отвечает за несколько ключевых задач. Он позволяет включать режим редактирования, отслеживать и обновлять время аренды, рассчитывать стоимость аренды и обрабатывать завершение аренды.Этот класс содержит свойства, такие как RentalStartTime (время начала аренды), RentDuration (продолжительность аренды), SelectedRate (выбранный тариф) и TimeLeft (оставшееся время аренды).Основные функции класса:

* Редактирование: Переключение между режимами редактирования и просмотра. Пользователь может включить режим редактирования, изменить данные аренды и подтвердить или отменить изменения;
* Удаление компьютера: С подтверждением и обработкой ошибок.
* Подтверждение изменений: Сохранение новой информации об аренде, расчет стоимости аренды на основе выбранного тарифа и добавление дохода в систему;
* Расчет оставшегося времени: Таймер обновляет оставшееся время аренды и обрабатывает завершение аренды, удаляя старую запись;
* Взаимодействие с данными: Класс использует различные репозитории для получения, обновления и удаления данных о компьютерах, тарифах, арендах и доходах.

**Выводы по разделу:**

В разделе описан процесс успешной реализации программного средства спроектированного в предыдущий разделах. Были описаны классы и методы этих классов которые задавали работу программного средства.

## 5. Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1. Тестирование авторизации

При авторизации программное средство учитывает многие факторы. Одним из таких является проверка корректности заполнения полей, так например поля логина и пароля не могут быть пустыми и не могут содержать информацию более 50 символов. Пример попытки ввода некорректных данных представлен на рисунках 5.1 и 5.2.

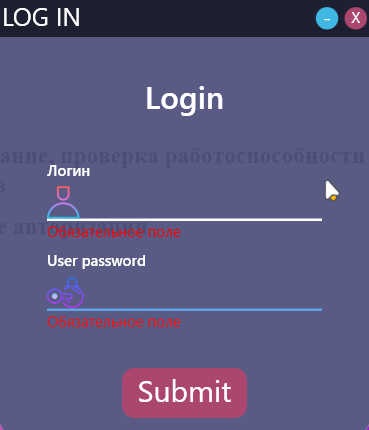


Рисунок 5.1 – Обработка пустых полей

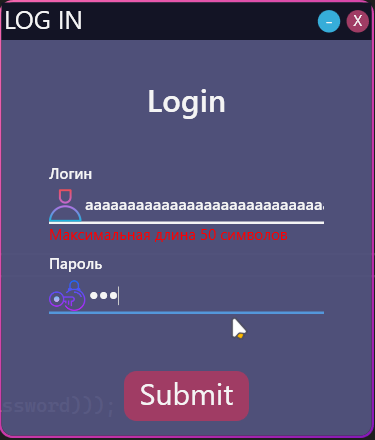


Рисунок 5.2 – Обработка переполненых полей

Таким образом программное средство подсказывает пользователю, что ему необходимо изменить чтобы успешно войти в систему.

## 5.2. Тестирование создания тарифа

В форме создания тарифа применяются следующие правила валидации:

* Все поля обязательны к заполнению;
* Ограничение на длину – 50 символов;
* Только неотрицательные числа в поле цена.

Пример тестирования системы валидации данной формы приведен на рисунке 5.3.

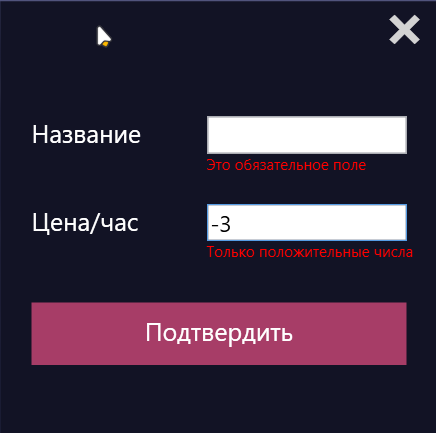


Рисунок 5.3 – Обработка полей создания тарифа

Таким образом, пользователь не сможет указать отрицательную стоимость аренды, так как такое поведение подозрительно и, предположительно, не соответствует ожидаемому результату.

## 5.3. Тестирование удаления тарифа

Пользователь не может удалить тарифы с которыми связаны компьютеры, так как это операция над данными не является безопасной. В этом случае программное средство будет выводить сообщение об ошибке, которое представлено на рисунке 5.4.

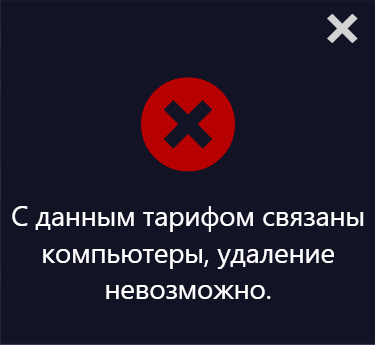


Рисунок 5.4 – Обработка удаления тарифа

Благодаря такого рода проверке, программное средство гарантирует сохранность данных, а также улучшает безопасность.

## 5.4. Тестирование добавления сотрудника

Пользователь должен соблюдать множество правил при заполнении данной формы. К ним относиться:

* Заполнить все поля;
* Минимальная длина каждого поля 3 символа;
* Максимальная длина каждого поля 50 символов;
* Минимальная длина пароля 8 символов;
* Пароль должен содержать хотя бы одну цифру и одну букву;
* Должен быть выбран хотя бы 1 клуб, при роли Admin.

В случае не соблюдения хотя бы одного правила, программное средство будет выводить сообщение об ошибке, которое представлено на рисунке 5.5.

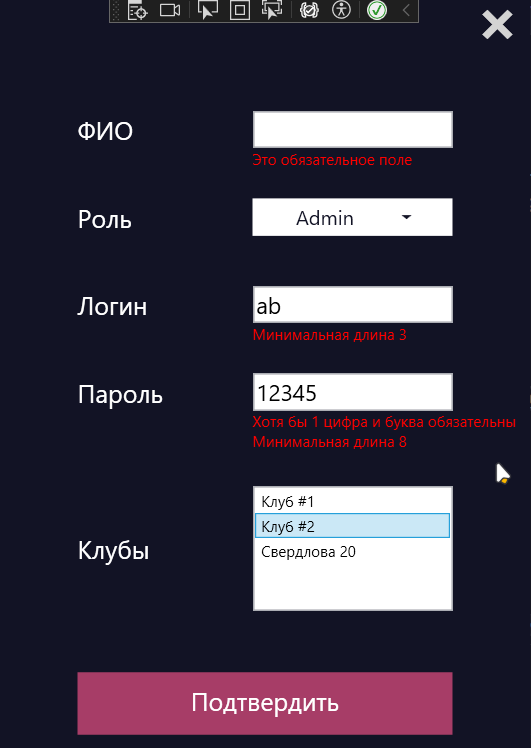


Рисунок 5.5 – Обработка полей добавления сотрудника

## Выводы по разделу:

В разделе описан процесс тестирования продукта. Были проверены случаи негативного и положительного тестирования. Были проверены все случаи валидации вводимых пользователем данных.

## Руководство по установке и использованию

При запуске программного средства пользователь попадает на страницу авторизации, содержащая форму входа.

Стартовая страница позволяет зарегистрированному пользователю совершить вход в аккаунт, для этого ему необходимо ввести корректные логин и пароль, после чего пользователь будет перенаправлен на главную страницу приложения.

Окно авторизации представлено на рисунках 6.1.

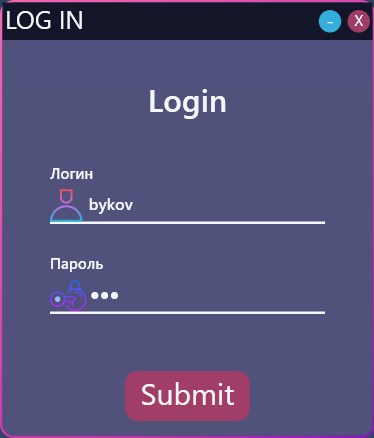


Рисунок 6.1 – Окно входа

Находясь на главной странице пользователь получает доступ к статистике текущего клуба. Текущий клуб можно сменить в верхнем меню при помощи соответсвующего комбобокса. Стоит отметить, что внешний вид приложения слегка отличается в зависимости от роли пользователя. На рисунках 6.2 и 6.3 представлены внешний вид главной страницы для администратора и владельца соответсвенно.

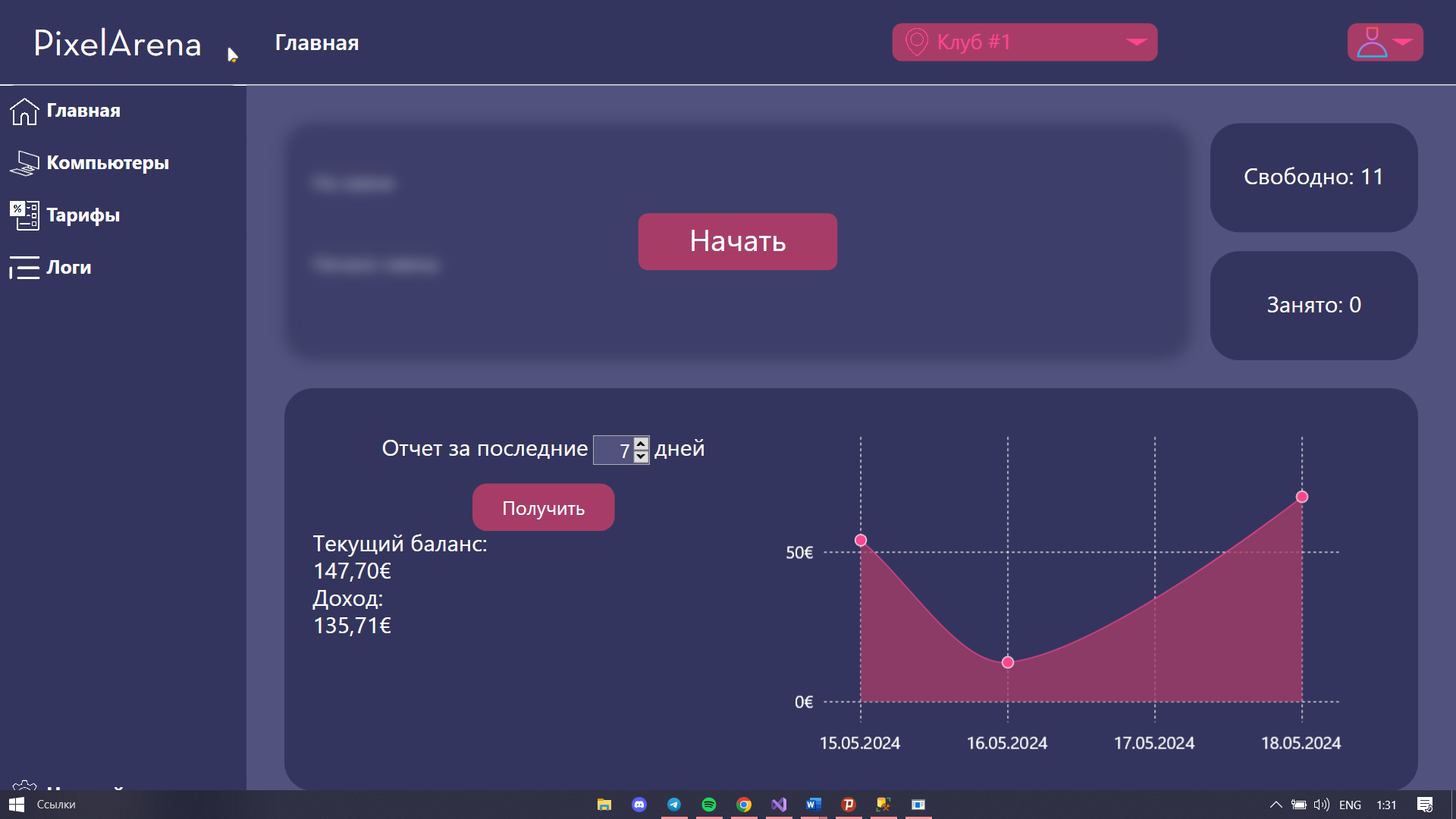


Рисунок 6.2 – Главная страница администратора

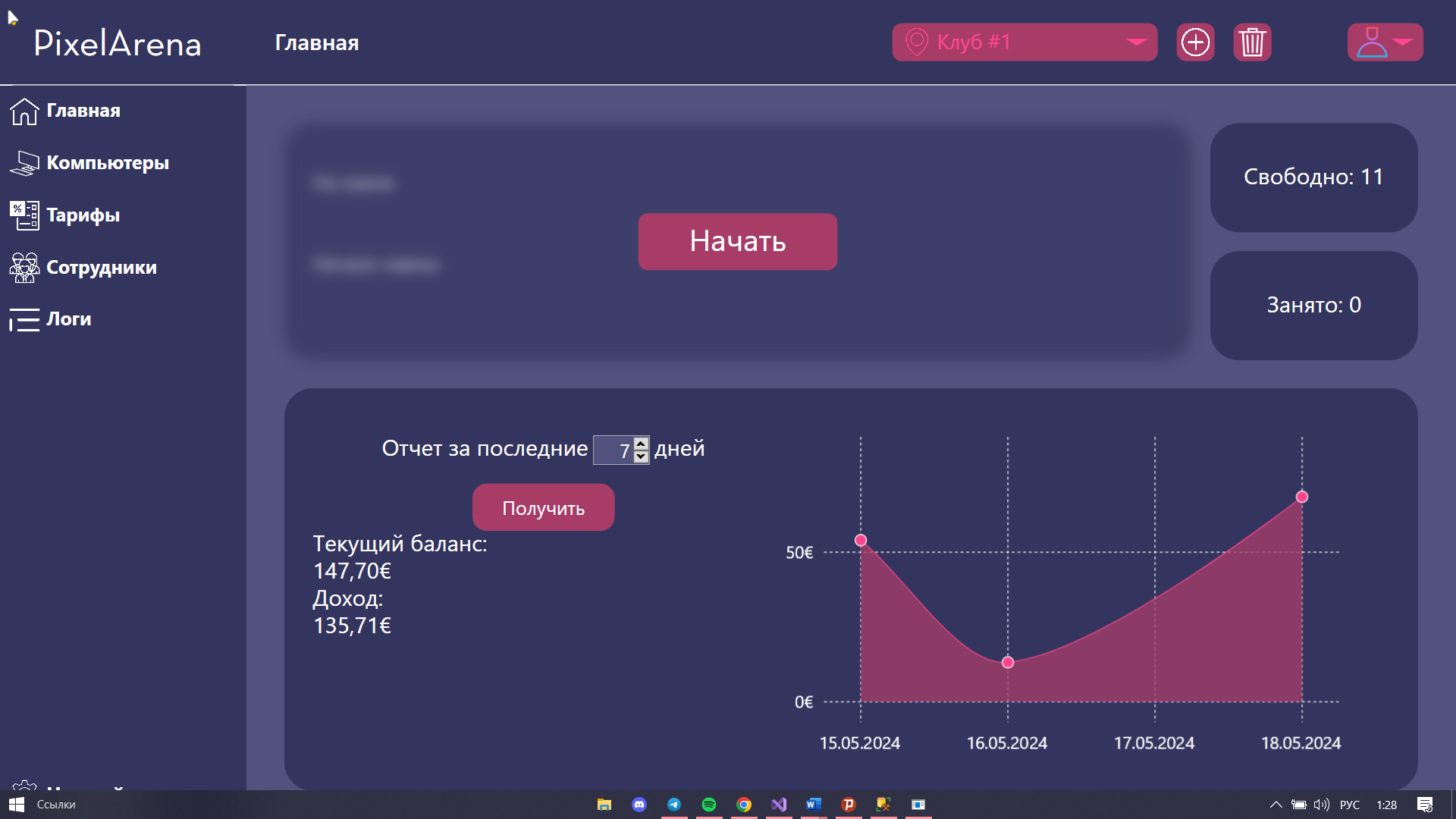


Рисунок 6.3 – Главная страница владельца

По центру экрана располагается большая кнопка начать. При её нажатии начинается новая сессия (смена), таким образом, до тех пор пока экземпляр приложения запущен, другие пользователи смогут увидеть, кто в данный момент находиться на смене в этом клубе.

К списку статистических данных предоставляемых на данной странице относиться текущий баланс клуба, доход за выбранный период, а так же график доходов построенный за выбранный период. Кроме того, на странице расположена статистика, сколько на данный момент, в текущем клубе, свободных и занятых компьютеров.  
 При помощи бокового меню пользователь может осуществлять навигацию по страницам.   
 Находясь на странице тарифы пользователь может добавлять, изменять и удалять тарифы. Страница изображена на рисунке 6.4.

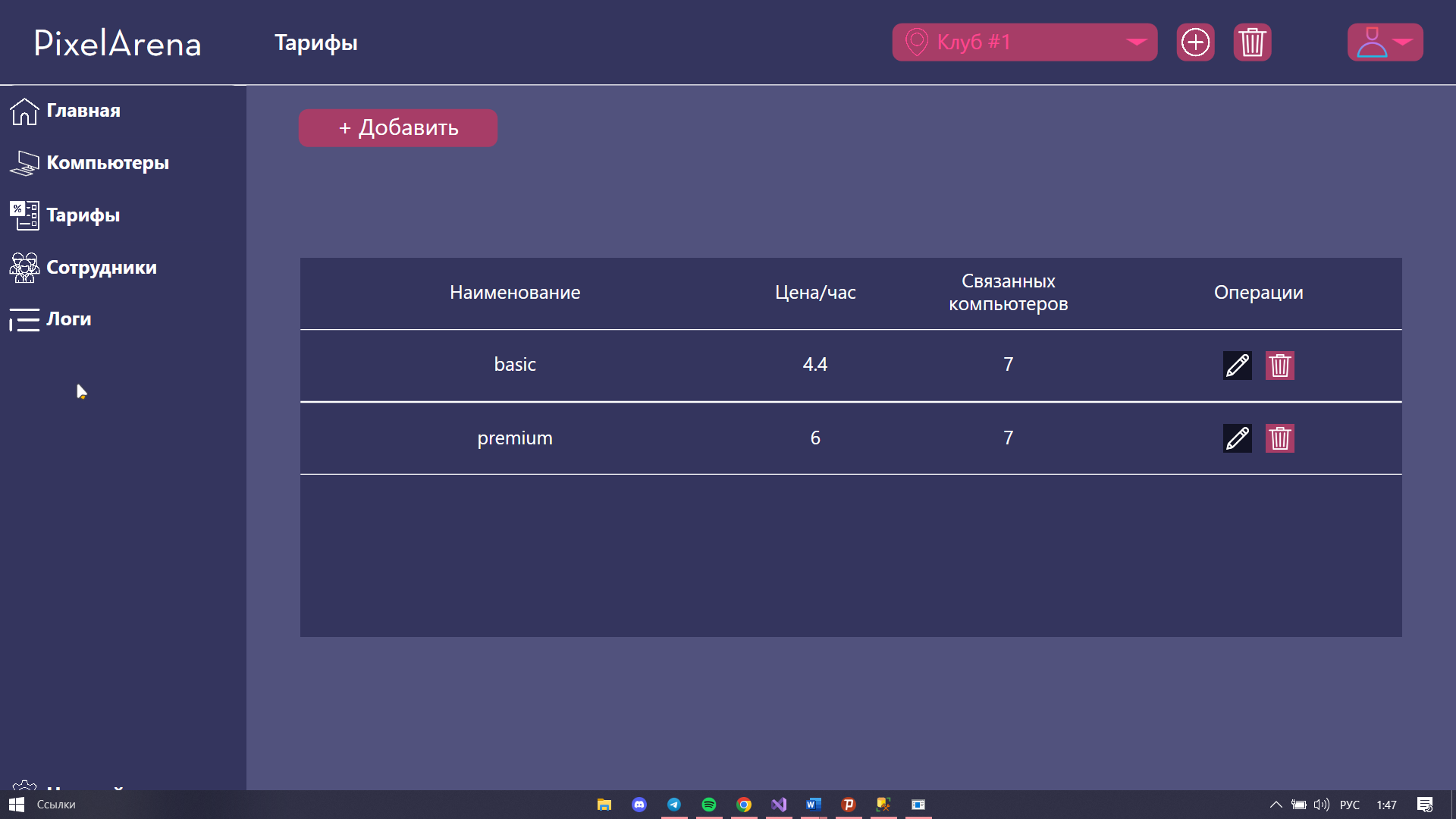


Рисунок 6.4 – Страница тарифы

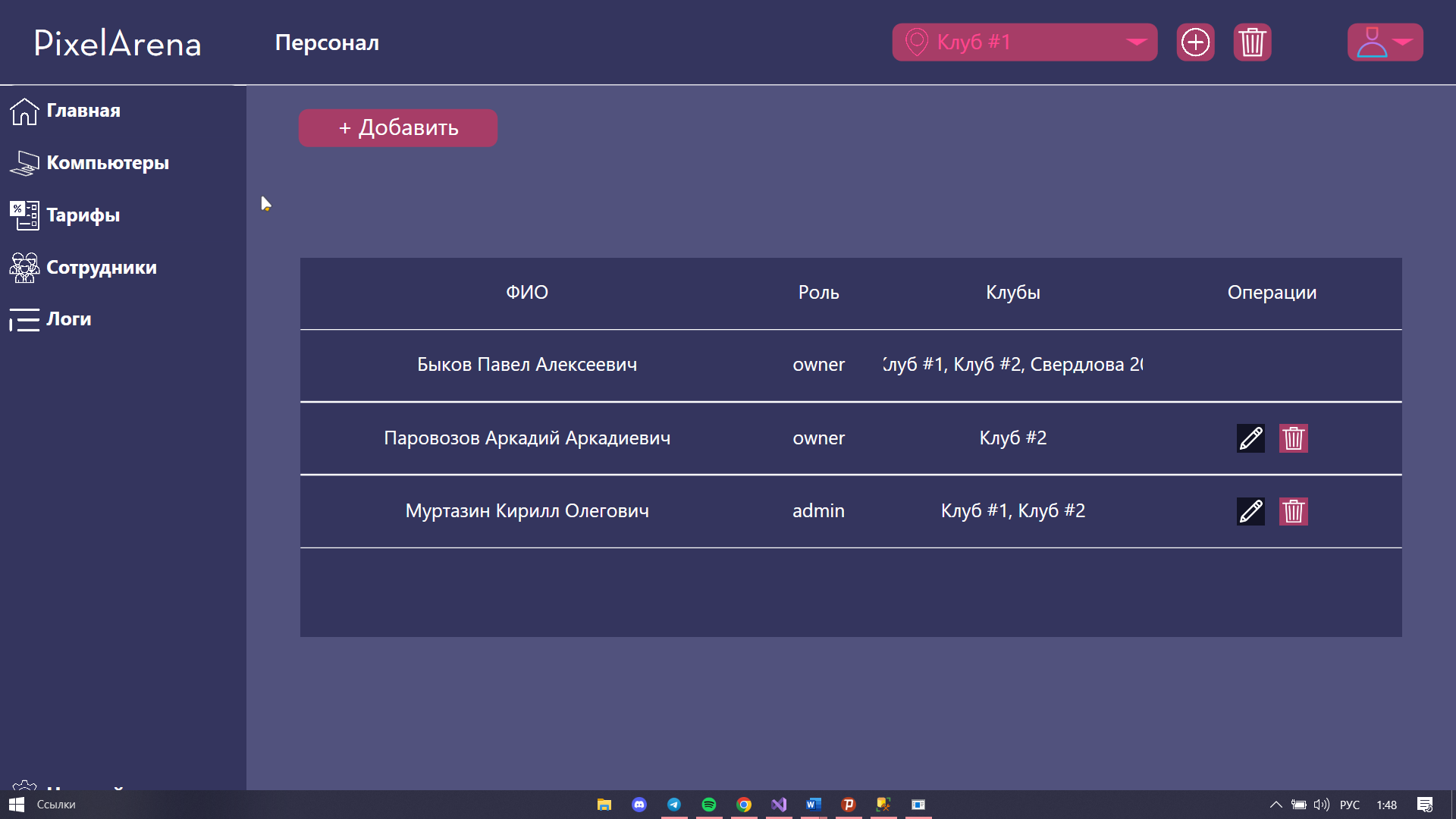
Находясь на странице сотрудники, которая доступная только для пользователей с ролью owner, пользователь может добавлять, удалять и изменять пользователей. Однако, пользователю запрещается изменять или удалять собственный аккаунт. Страница изображена на рисунке 6.5.  


Рисунок 6.5 – Страница сотрудники

Находясь на странице логи пользователь получает текстовую информацию о всех действиях совершенным им в системе. Данная информация сохраняться при выходе пользователя из приложения, однако сбрасывается после полного закрытия приложения. Также информацию можно экспортировать в файл с расширением .log, путь к файлу можно изменить в настройках, доступ к которым предоставляется по нажатию на соотвествующию кнопку в боковом меню. Страница логи представлена на рисунке 6.6.

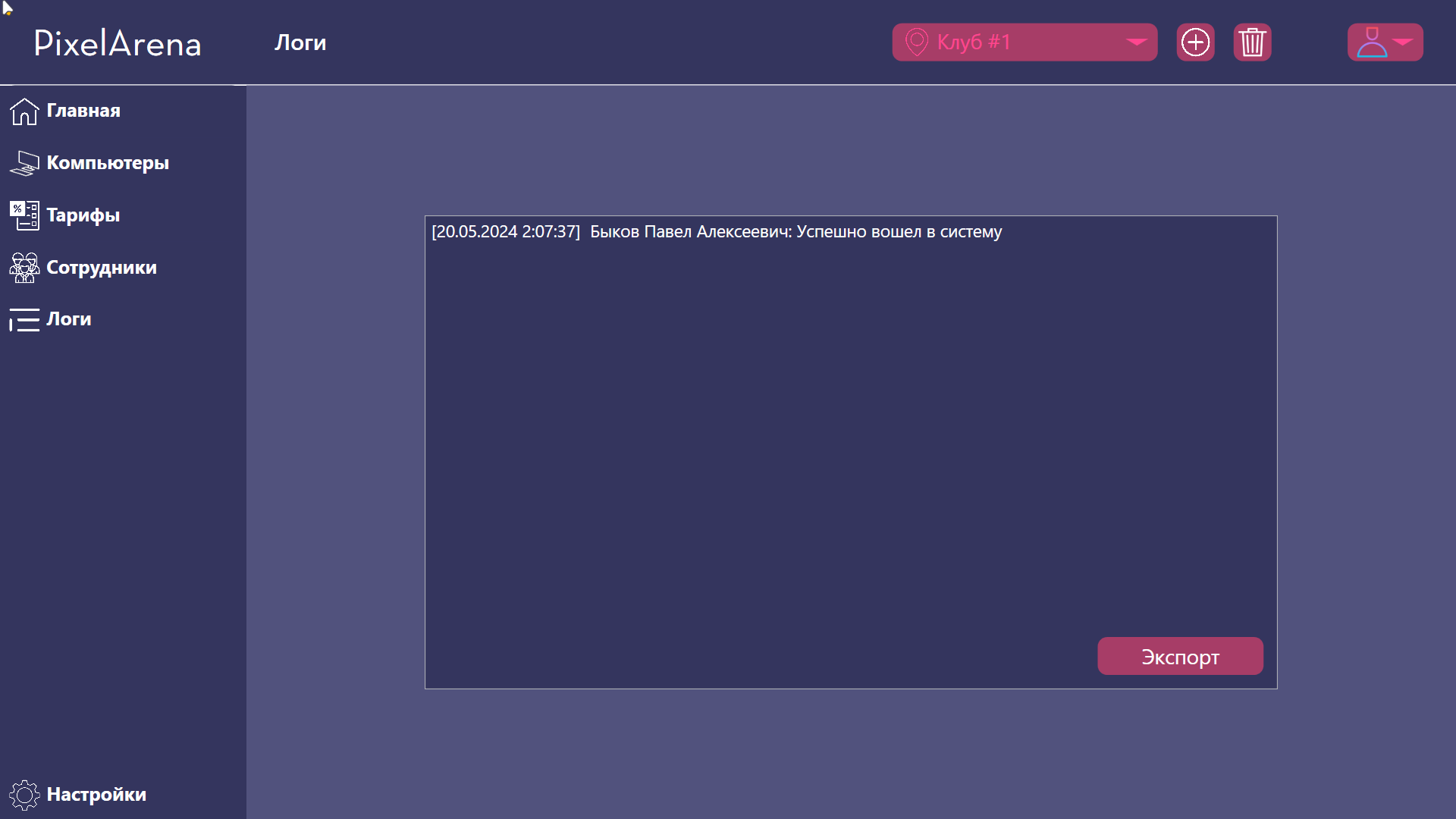


Рисунок 6.6 – Страница логи

Находясь на странице компьютеры, пользователь может добавлять компьютеры. Компьютеры на данной странице представленны в виде карточке с соответсвующей информацией о номере компьютера, тарифе и статусе. Страница компьютеры проиллюстрирована на рисунке 6.7.



Рисунок 6.7 – Страница компьютеры

Кроме того, нажав на карточку любого компьютера, открывается дополнительно окно, позволяющие осуществлять детальное изменение компьютера и связанной с ним арендой. Это окно изображено на рисунке 6.8.

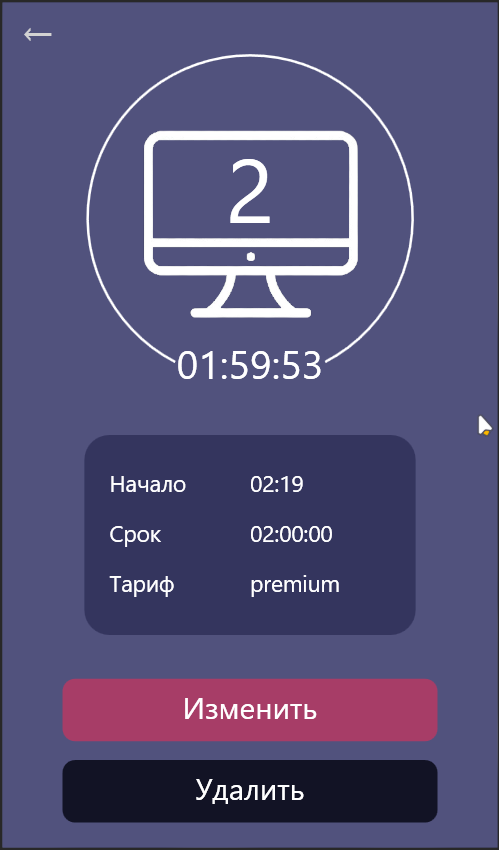


Рисунок 6.8 – Окно деталей компьютера

По нажатию кнопки “Удалить”, после подтверждения, будет произведено удаление компьютера. По нажатию кнопки “Изменить”, внешний вид измениться, поля станут редактируемыми и можно будет изменить детали аренды и/или компьютера. При сохранении изменений выведиться сумма которая должна быть взыскана с клиента, сумма автоматически прибавляется на баланс клуба и заноситься в статистику.

## Выводы по разделу:

В разделе описан процесс использования программного средства. В нем пользователя инструктирую, как воспользоваться каждой из функции, которые предоставляет программное средство.

## Заключение

В ходе выполнения курсового проекта было разработано программное средство для управления компьютерным клубом на языке C# с использованием технологий Entity Framework Core, WPF.

При разработке программного средства были выполнены все пункты из указанного списка предполагаемого основного функционала приложения, а именно:

* создана база данных;
* реализована возможность авторизации и создания новых аккаунтов;
* реализована возможность управлением тарифами;
* реализована возможность управлением компьютерами;
* реализована возможность управлением арендами;
* реализована автоматического получения доходной статистики;
* реализована генерация логов;
* реализована возможность экспорта логов в файл.

Тестирование программы показало, что она работает корректно и выполняет все свои функции.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список литературы**

1. Прайс М., C# 9 и .NET 5. Разработка и оптимизация – Петербург 2022. – 832с.
2. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
3. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 28.03.2024
4. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 23.04.2024

## ПРИЛОЖЕНИЕ А

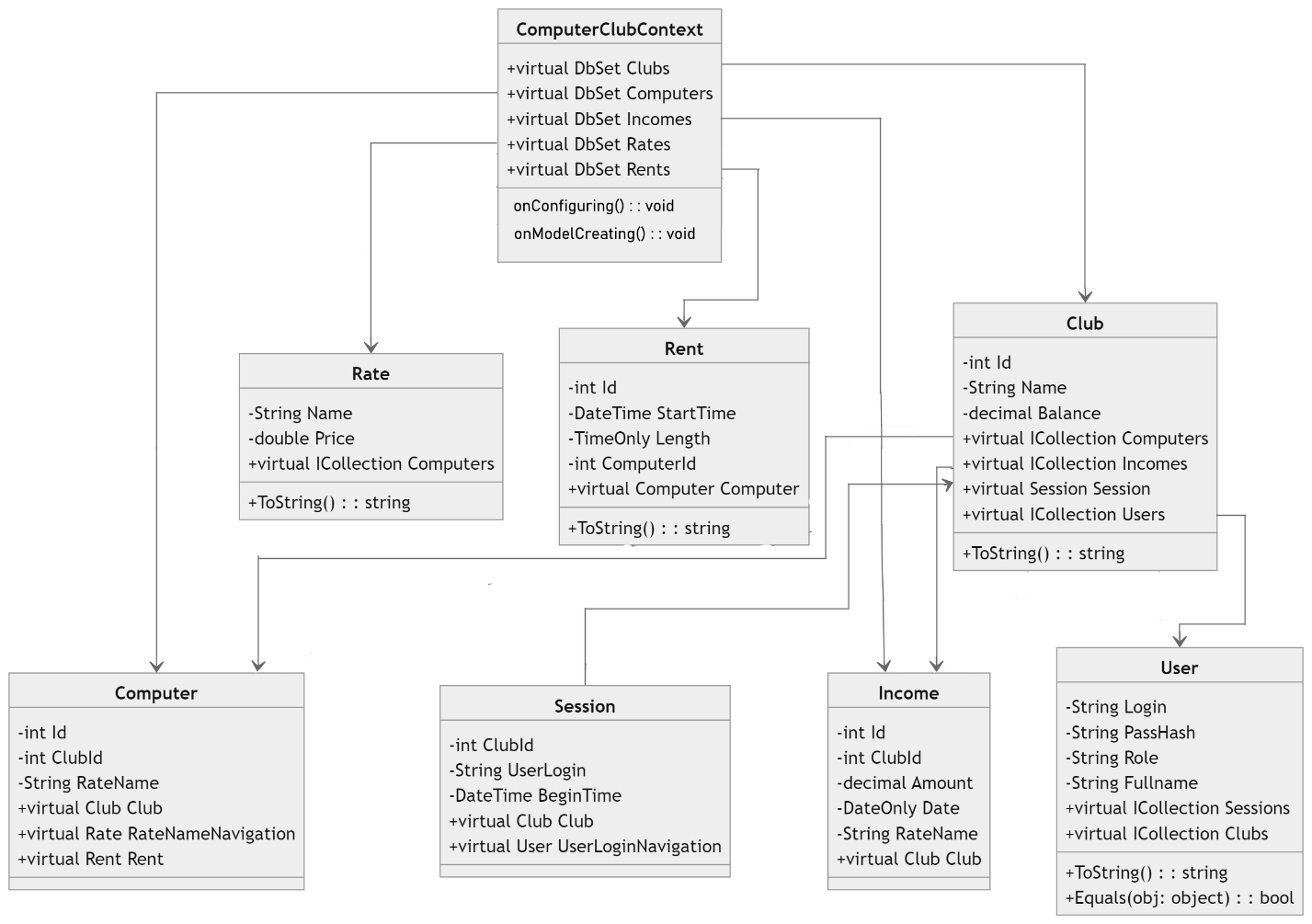


Рисунок А.1 – диаграмма классов

## ПРИЛОЖЕНИЕ Б

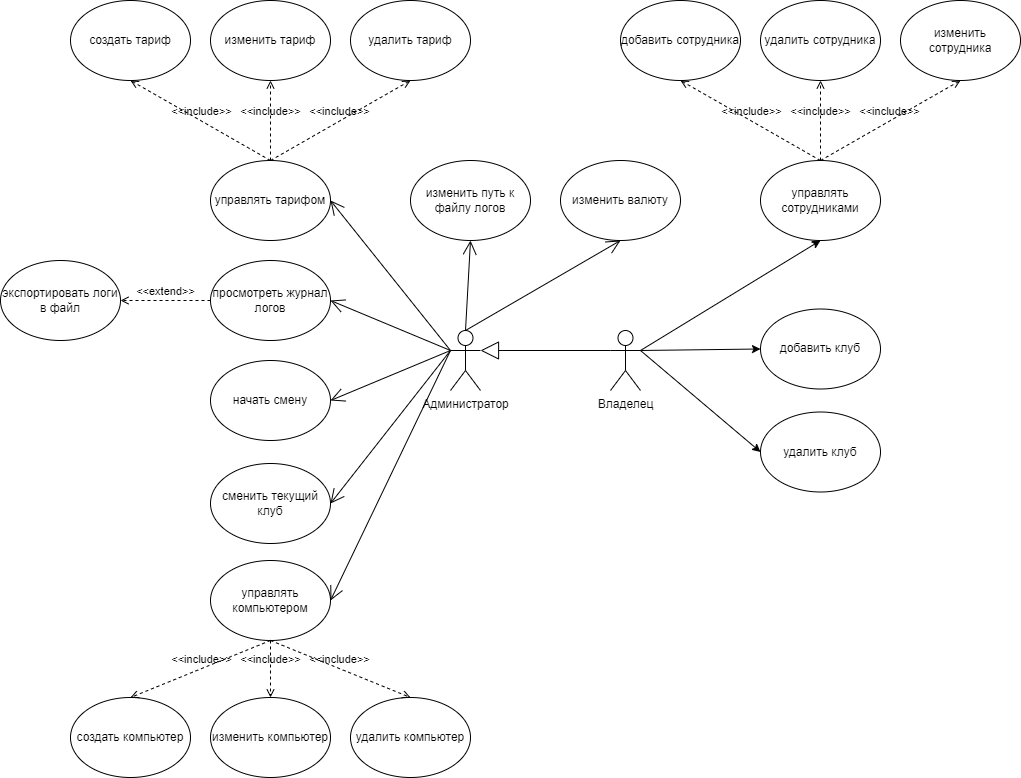


Рисунок Б.1 – диаграмма вариантов использования

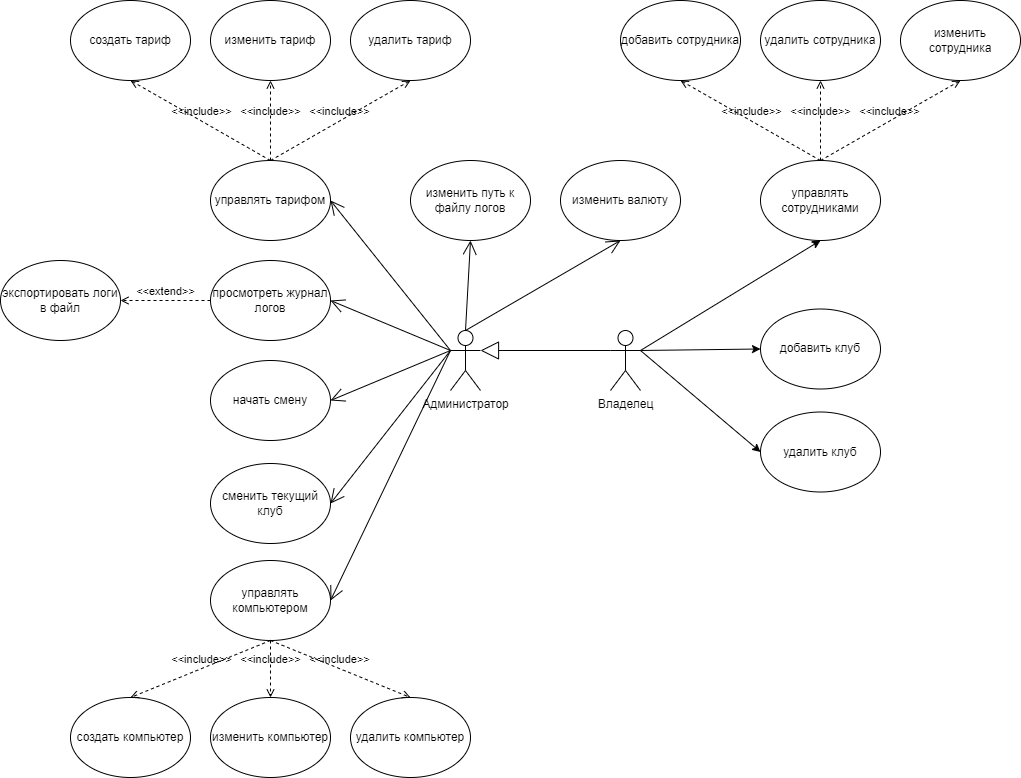
ПРИЛОЖЕНИЕ В

Рисунок В.1 – диаграмма последовательности

## ПРИЛОЖЕНИЕ Г

Листинг 1: Класс AuthService

public partial class AuthService : ObservableObject

{

public enum Roles {Admin, Owner}

private static AuthService \_instance;

private readonly ComputerClubContext \_context;

[ObservableProperty]

private User \_currentUser;

[ObservableProperty]

private Club \_currentClub;

private AuthService(ComputerClubContext context)

{

\_context = context;

\_context.Sessions.Load();

\_context.Clubs.Load();

\_context.Users.Load();

}

public static AuthService GetInstance(ComputerClubContext context = null)

{

try

{

if (\_instance == null)

{

if(context == null)

{

throw new AuthException("Ошибка прав доступа: контекста не существует");

}

\_instance = new AuthService(context);

}

return \_instance;

}

catch (Exception e)

{

NotifyModalWindow.Show(NotifyModalWindow.NotifyKind.Error, "Ошибка получения authservice: " + e.Message);

}

return null;

}

public static string GetHash(string pass)

{

MD5 md5 = MD5.Create();

byte[] passBytes = Encoding.ASCII.GetBytes(pass);

byte[] passHash = md5.ComputeHash(passBytes);

return Convert.ToHexString(passHash);

}

public bool TryAuth(string login, string password)

{

var temp = \_context.Users.Include(s => s.Clubs).ToList();

var temp1 = \_context.Clubs.Include(c => c.Users).ToList();

User user = \_context.Users.FirstOrDefault(u => u.Login == login);

if (user == null)

{

return false;

}

string passHash = GetHash(password);

if(!user.PassHash.TrimEnd().Equals(passHash))

{

return false;

}

// данные совпали

CurrentUser = user;

CurrentClub = user.Clubs.First();

Logger.Add("Успешно вошел в систему");

return true;

}

public ICollection<Club> GetAvailableClubs()

{

return CurrentUser?.Clubs;

}

public void Clear()

{

CurrentUser = null;

CurrentClub = null;

}

}