

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра \_\_\_\_\_ **Вычислительной техники** \_\_\_\_\_  
(полное название кафедры)

Утверждаю

Зав. кафедрой \_\_\_\_\_ **ВТ** \_\_\_\_\_

\_\_\_\_\_ **к.т.н., Якименко А.А.** \_\_\_\_\_  
(подпись, инициалы, фамилия)

«\_\_\_» \_\_\_\_\_ 202**0** г.

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

\_\_\_\_\_ **Дунаева Никиты Юрьевича** \_\_\_\_\_  
(фамилия, имя, отчество студента – автора работы)

\_\_\_\_\_ **Разработка Desktop-приложения для автоматизации учёта клиентопотока и движения** \_\_\_\_\_  
(тема работы)

\_\_\_\_\_ **денежных средств в сети спортивных центров** \_\_\_\_\_

\_\_\_\_\_ **Факультет автоматики и вычислительной техники** \_\_\_\_\_  
(полное название факультета)

Направление подготовки **09.03.01 Информатика и вычислительная техника** \_\_\_\_\_  
(код и наименование направления подготовки бакалавра)

**Руководитель  
от НГТУ**

\_\_\_\_\_ **Мищенко П. В.** \_\_\_\_\_  
(фамилия, имя, отчество)

\_\_\_\_\_ **ст. препод. каф. ВТ, НГТУ** \_\_\_\_\_  
(ученая степень, ученое звание)

\_\_\_\_\_ (подпись, дата)

**Автор выпускной  
квалификационной работы**

\_\_\_\_\_ **Дунаев Н. Ю.** \_\_\_\_\_  
(фамилия, имя, отчество)

\_\_\_\_\_ **АВТФ, АВТ-610** \_\_\_\_\_  
(факультет, группа)

\_\_\_\_\_ (подпись, дата)

Новосибирск 202**0**

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра \_\_\_\_\_ Вычислительной техники  
(полное название кафедры)

УТВЕРЖДАЮ

Зав. кафедрой к.т.н., Якименко А.А.  
(фамилия, имя, отчество)

\_\_\_\_\_  
(подпись, дата)

**ЗАДАНИЕ  
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА**

студенту \_\_\_\_\_ Дунаеву Никите Юрьевичу  
(фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника  
(код и наименование направления подготовки бакалавра)

\_\_\_\_\_  
Факультет автоматизации и вычислительной техники  
(полное название факультета)

Тема Разработка Desktop-приложения для автоматизации учёта клиентопотока и  
(полное название темы выпускной квалификационной работы бакалавра)  
движения денежных средств в сети спортивных центров

Исходные данные (или цель работы) Исходные данные: язык гипертекстовой  
разметки HTML, язык программирования JavaScript, таблицы стилей CSS, браузер  
Google Chrome, JavaScript-библиотеки Vue, Electron, Knex, Express,  
JavaScript-транслятор Node, система управления базами данных PostgreSQL,  
редактор исходного кода Visual Studio Code, система контроля версий Git.

Цель работы: разработка и внедрение приложения для автоматизации учета  
клиентопотока и движения денежных средств в сети спортивных центров  
«CHAMPION GYM».

Структурные части работы \_\_\_\_\_

Введение

Постановка задачи

Обзор текущей системы учета «CHAMPION GYM»

Анализ средств разработки Desktop-приложений

Анализ средств разработки Back-End

Формирование требований

Выбор средств реализации

Проектирование

Реализация

Тестирование

Внедрение

Заключение

Список использованных источников

Приложение А

Приложение Б

Задание согласовано и принято к исполнению.

**Руководитель  
от НГТУ**

Мищенко П. В.

(фамилия, имя, отчество)

ст. препод. каф. ВТ, НГТУ

(ученая степень, ученое звание)

\_\_\_\_\_  
(подпись, дата)

**Студент**

Дунаев Н. Ю.

(фамилия, имя, отчество)

АВТФ, АВТ-610

(факультет, группа)

\_\_\_\_\_  
(подпись, дата)

Тема утверждена приказом по НГТУ № 1256/2 от « 02 » марта 2020 г.  
изменена приказом по НГТУ № \_\_\_\_\_ от « \_\_\_\_ » \_\_\_\_\_ 2020 г.

ВКР сдана в ГЭК № \_\_\_\_\_, тема сверена с данными приказа

\_\_\_\_\_  
(подпись секретаря государственной экзаменационной комиссии по защите ВКР, дата)

\_\_\_\_\_  
(фамилия, имя, отчество секретаря государственной  
экзаменационной комиссии по защите ВКР)

**Примечание:** поля данного блока заполняются секретарем ГЭК при приеме ВКР к защите.

## Реферат

Выпускная квалификационная работа выполнена студентом Дунаевым Никитой Юрьевичем.

Место создания – ФГБОУ ВО «Новосибирский государственный технический университет».

Руководитель – старший преподаватель кафедры Вычислительной техники, Мищенко П. В.

Пояснительная записка содержит: 9 глав, 111 страниц, 51 рисунок, 7 таблиц, 2 приложения, 15 источников литературы.

Данная работа содержит в себе обзор текущей системы учета, формирование требований, анализ средств разработки и выбор средств реализации, проектирование, реализацию и тестирование новой системы, а также стратегию ее внедрения.

Значимость данной работы заключается в том, что созданная система учета позволяет автоматизировать деятельность администратора спортивного центра, предоставляет возможность анализа эффективности предприятия, а также расширяет спектр предоставляемых сетью услуг.

В результате проделанной работы было реализовано и внедрено приложение, автоматизирующее учет клиентопотока и движения денежных средств.

Ключевые слова: desktop-приложение, Node, Vue, Electron, Express, PostgreSQL, API, система учета.

## Оглавление

Введение.....	7
Постановка задачи.....	8
1 Обзор текущей системы учета «CHAMPION GYM».....	9
2 Анализ средств разработки Desktop-приложений.....	16
3 Анализ средств разработки Back-End.....	20
4 Формирование требований .....	25
4.1 Требования к дизайну .....	25
4.2 Функциональные требования .....	25
4.3 Требования к хранению данных .....	27
4.4 Требования к ПО .....	27
4.5 Требования к безопасности .....	27
5 Выбор средств реализации.....	28
5.1 Клиентская логика.....	28
5.2 Серверная логика.....	30
6 Проектирование .....	34
6.1 Проектирование клиента .....	34
6.2 Проектирование сервера.....	35
7 Реализация .....	48
7.1 Реализация сервера.....	48
7.2 Реализация клиента .....	54
8 Тестирование .....	80
9 Внедрение .....	102

Заключение .....	103
Список используемых источников .....	104
Приложение А .....	106
Приложение Б .....	111

## **Введение**

На момент написания выпускной квалификационной работы в городе Новосибирске существует большое количество конкурирующих спортивных центров. Для того, чтобы сохранить позиции на рынке, центры стараются предоставлять клиентам новые услуги, начиная с огромного выбора групповых занятий, заканчивая возможностью приобретения сетевых абонементов. Рост количества услуг предполагает возникновение потребности в современном программном обеспечении и средствах автоматизации. С ростом количества услуг приток клиентов увеличивается, вследствие чего появляются большие объемы клиентских данных, которые необходимо хранить и обрабатывать, используя те же программные решения.

На сегодняшний день на рынке существует множество программных и аппаратных средств автоматизации, но далеко не все из них соответствуют требованиям компаний. В рамках данной выпускной квалификационной работы будет рассмотрена разработка системы учета клиентопотока и движения средств в сети спортивных центров «CHAMPION GYM». В процессе расширения предприятия постоянно появляются новые услуги, возникает необходимость обслуживать большее количество клиентов, появляется потребность в качественном и надежном приложении, отвечающем требованиям удобства, быстродействия и безопасности.

На данный момент компания использует универсальную систему учета «USU». Программа предоставляет возможность ведения учета клиентов и покрывает практически все потребности сети спортивных центров, однако имеет несколько ключевых недостатков:

- избыточный перегруженный функционал, большая часть которого не используется;
- отсутствие возможности вести учет клиентов сети с единой базой;

- отсутствие возможности составления отчета, соответствующего специфике конкретного предприятия.

Новая система учета позволит автоматизировать деятельность администратора сети спортивных центров, предоставит возможность просмотра сетевого отчета для анализа продуктивности отдельных тренеров и эффективности предприятия в целом, а также даст возможность клиентам владеть сетевым абонементом. Обновленный функционал позволит предприятию избавиться от ограничений, создаваемых старой системой учета и составить конкуренцию другим спортивным залам, предоставив клиенту новый спектр услуг.

### **Постановка задачи**

Целью данной выпускной квалификационной работы является разработка и внедрение приложения для автоматизации учета клиентопотока и движения денежных средств в сети спортивных центров «CHAMPION GYM».

Задачи, подлежащие выполнению в рамках выпускной квалификационной работы:

- провести анализ текущей системы учета, выявить преимущества и недостатки;
- сформировать требования к новой системе учета и обосновать выбор технологий;
- спроектировать и разработать базу данных для системы учета;
- спроектировать и разработать серверную и клиентскую логику приложения;
- провести функциональное тестирование desktop-приложения;
- произвести апробацию и внедрение программного продукта.



## 1 Обзор текущей системы учета «CHAMPION GYM»

Так как сеть спортивных центров «CHAMPION GYM» уже использует приложение для автоматизации учёта клиентопотока и движения денежных средств «USU», проектировать новое приложение стоит, обращая внимание на преимущества и недостатки старого [1]. Ниже представлено описание используемого функционала текущей системы учета.

1) Окно для ввода номера абонемента представлено на рисунке 1.

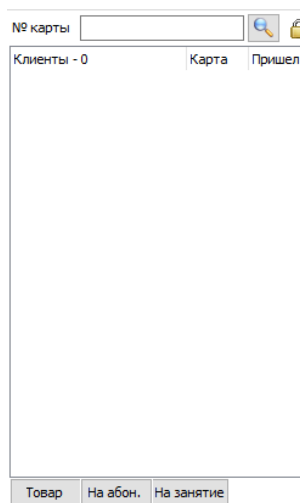


Рисунок 1 - Окно для ввода номера абонемента до прихода клиента

2) Окно с выводом найденных абонементов, фотографией и количеством занятий, а также с кнопкой «Отметить» для списания занятий представлено на рисунке 2.

Предварительная запись		
№	Помещение	Время
1		

Отметить	Не отмечать	На абон.	На клиента
----------	-------------	----------	------------

ФИО **Мирослав**  
 Курс **Дансeтix 8 занятий | 1600**  
 Статус **Открыт**  
 Дата начала **20.11.2019**  
 Окончание **20.12.2019**  
 Всего занятий **16**  
 Ост. занятий **8**  
 Ост. гостевых **0**  
 Ост. заморозки **0**  
 Ост. оплатить **1 600,00**  
 Примечание **На следующей тренировке**




Рисунок 2 - Окно для ввода номера абонемента после прихода клиента

3) Раздел с людьми (экран горизонтально делится пополам, в верхней части — таблица с людьми, в нижней части — фото по каждому из людей и количество приглашенных им) представлен на рисунке 3.

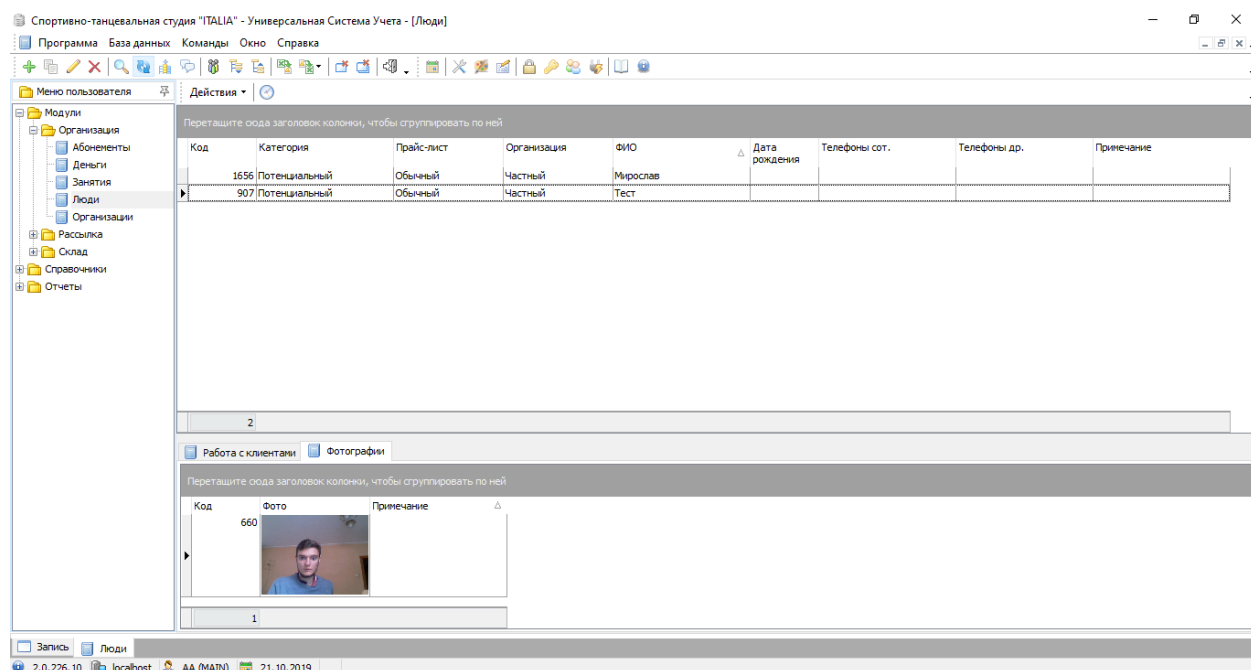


Рисунок 3 - Раздел с людьми

4) Раздел с абонементом (экран горизонтально делится пополам, в верхней части — таблица с абонементом, в нижней части — оплаты по каждому из абонементов) представлен на рисунке 4.

Код	Статус	Номер карты	ФИО	Вид абонента	У сотрудника	Дата начала	Дата окончания	Всего занятий	К оплате	Осталось занятий	Ожидаемая оплата	Гостевых оплат	Осталось записки	Осталось заморозки	Примечание
2410	Открыт	2	Миронов	Дипломат 8 занятий 1 1600	Челубеева Елена	20.11.2019	20.12.2019	16,00	3 200,00	8,00	1 600,00	0,00	0,00	0,00	Не оплачено
2409	Открыт	1	Тест	Абонемент на массаж		07.12.2017		30,00	0,00	30,00	0,00	0,00	0,00		

Код	Дата	Касса	Вид ставки	Ставка преподавателя	Сумма преподавателя	Сумма
3208	20.10.2019	Наличными	Процент от суммы	40,00	640,00	1 600,00

Рисунок 4 - Раздел с абонементом

5) Окно добавления нового клиента представлено на рисунке 5. В соответствующие поля формы заносится личная информация о человеке.

Людям	
Категория *	Потенциальный
Прайс-лист *	Обычный
Организация *	Частный
ФИО *	
Дата рождения	
Пол *	Ж
Вес	0,00
Рост	0,00
Место работы	
Должность	
Телефоны сот.	
Телефоны др.	
E-mail	
Адрес	
Документы	
Получать рассылку	<input checked="" type="checkbox"/>
Примечание	

Рисунок 5 - Окно добавления/редактирования клиента

6) Окно добавления нового абонента представлено на рисунке 6. В соответствующие поля формы заносится информация о приобретаемом человеком абоненте.

The screenshot shows a web form titled 'Абоненты' (Subscribers). The form contains the following fields:

Абоненты	
Номер карты	
ФИО *	
Вид абонента *	
У сотрудника	
В группе	
Время начала	
Дата начала *	21.10.2019
Дата окончания	
Сумма скидки *	0,00
Валюта *	RUB
Источник информации *	Не указано
Примечание	

At the bottom of the form, there are two buttons: 'Сохранить' (Save) with a green checkmark icon and 'Отменить' (Cancel) with a red X icon.

Рисунок 6 - Окно добавления/редактирования абонента

7) Окно для внесения новой оплаты в разделе «Абоненты» представлено на рисунке 7.

The screenshot shows a web form titled 'Оплаты' (Payments). The form contains the following fields:

Оплаты	
Дата *	21.10.2019
Касса *	
Сумма *	0,00
Примечание	

At the bottom of the form, there are two buttons: 'Сохранить' (Save) with a green checkmark icon and 'Отменить' (Cancel) with a red X icon.

Рисунок 7 - Окно внесения/редактирования оплаты

8) Окно «Виды абонементов» представлено на рисунке 8. Также окно редактирования вида абонемента представлено на рисунке 9.

Курс									
Код	Наименование	Стоимость	Занятий	Гостевых посещений	Длительность, мин	Срок заморозки, дней	Время посещения от	Время посещения до	
Курс : Групповые занятия (8)									
244	Dancemix 8 занятий   1600	1 600,00	8,00	0,00	900	0,00	07:30:00		
245	Абонемент на массаж	0,00	30,00	0,00	900	0,00	07:30:00		
235	Аренда зала 7200 р.	7 200,00	30,00	0,00	900	0,00	07:30:00		
231	Детские танцы (15 занятий) 1500	1 500,00	15,00	0,00	60	0,00	07:30:00		
236	Разовое занятие	300,00	1,00	0,00	900	0,00	07:30:00		
243	Стрип и восток 8 занятий   2100	2 100,00	8,00	0,00	900	0,00	07:30:00		
242	Фитнес 12 занятий   1800	1 800,00	12,00	0,00	900	0,00	07:30:00		
241	Фитнес 8 занятий   1600	1 600,00	8,00	0,00	900	0,00	07:30:00		

Рисунок 8 - Раздел «Виды абонементов»

Рисунок 9 - Окно для редактирования/добавления вида абонемента

10) Окно для продления абонемента представлено на рисунке 10.

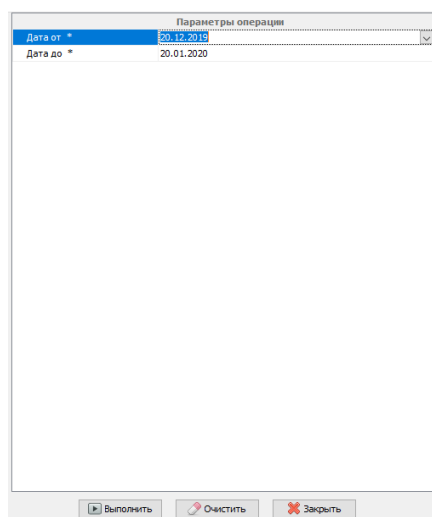


Рисунок 10 - Окно для продления абонемента

11) Прикрепление фотографии с веб-камеры представлено на рисунке 11.

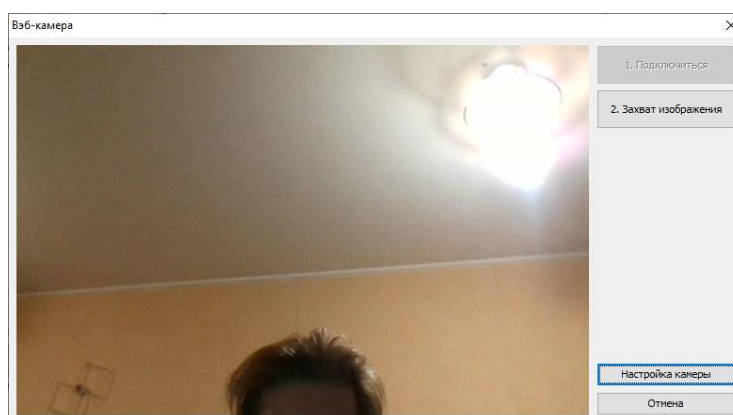


Рисунок 11 - Окно работы с веб-камерой

Текущая система учета имеет множество неиспользуемых и бесполезных вкладок, замедляющих работу с приложением, простоту в навигации и обучение новых сотрудников-администраторов, а также не предоставляет возможность учета клиентопотока в сети тренажерных залов [1].

Таким образом, новая система учета должна обладать всем используемым функционалом старой системы, перечисленным выше. Помимо этого, новая система будет предоставлять следующие возможности:

- возможность ведения учета клиентов сети тренажерных залов с единой базой
- возможность составления отчета, соответствующего специфике конкретной сети тренажерных залов

## **2 Анализ средств разработки Desktop-приложений**

### **Qt**

Qt — кроссплатформенный фреймворк для разработки программного обеспечения на языке программирования C++.

Qt позволяет запускать написанное с его помощью программное обеспечение в большинстве современных операционных систем путём простой компиляции программы для каждой системы без изменения исходного кода. Включает в себя все основные классы, которые могут потребоваться при разработке прикладного программного обеспечения, начиная от элементов графического интерфейса и заканчивая классами для работы с сетью, базами данных и XML. Является полностью объектно-ориентированным, расширяемым и поддерживающим технику компонентного программирования.

Отличительная особенность — использование метаобъектного компилятора — предварительной системы обработки исходного кода. Расширение возможностей обеспечивается системой плагинов, которые возможно размещать непосредственно в панели визуального редактора. Также существует возможность расширения привычной функциональности виджетов, связанной с размещением их на экране, отображением, перерисовкой при изменении размеров окна.

Одним из преимуществ проекта Qt является наличие качественной документации. Статьи документации снабжены большим количеством примеров. Исходный код самой библиотеки хорошо форматирован, подробно комментирован и легко читается, что также упрощает изучение Qt [10].

### **JavaFX**

JavaFX — платформа на основе Java для создания приложений с насыщенным графическим интерфейсом. Может использоваться как для



создания настольных приложений, запускаемых непосредственно из-под операционных систем, так и для интернет-приложений, работающих в браузерах, и для приложений на мобильных устройствах. JavaFX призвана заменить использовавшуюся ранее библиотеку Swing. Платформа JavaFX конкурирует с Microsoft Silverlight, Adobe Flash и аналогичными системами.

Компоненты:

- Средства разработки — компилятор и среда исполнения JavaFX, язык программирования JavaFX Script, а также графические, медийные и веб-библиотеки для создания RIA-приложений для настольных компьютеров, веб-сайтов и мобильных устройств.
- Интегрированная среда разработки NetBeans IDE — средства для кодирования и отладки приложений, написанных на JavaFX Script. В редакторе JavaFX Script есть возможность быстрого добавления объектов JavaFX с уже готовыми геометрическими фигурами, компонентами интерфейса пользователя, средствами преобразования и анимацией.
- Production Suite — набор инструментов и плагинов для импорта графических объектов в приложения JavaFX. Включает следующие компоненты:
  - Плагины для графических редакторов Adobe Photoshop CS3, CS4 и Adobe Illustrator CS3, CS4. С помощью плагинов можно экспортировать графические объекты из этих приложений в код JavaFX Script.
  - Media Factory: набор инструментов для конвертирования SVG-графики в код JavaFX и просмотра графических объектов, импортированных в JavaFX из других форматов. Также включает примеры приложений, учебные курсы, статьи, API-документацию и примеры кода [12].

**.NET**

.NET Framework — программная платформа, выпущенная компанией Microsoft в 2002 году. Основой платформы является общезыковая среда исполнения Common Language Runtime (CLR), которая подходит для разных языков программирования. Функциональные возможности CLR доступны в любых языках программирования, использующих эту среду.

Из преимуществ можно выделить:

- Поддержка нескольких языков. Основой платформы является общезыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.
- Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.
- Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.
- Разнообразие технологий. Общезыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий,

которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и UWP, для создания более простых графических приложений - Windows Forms. Для разработки мобильных приложений - Xamarin. Для создания веб-сайтов - ASP.NET и т.д [13].

## **Electron**

Electron — фреймворк, разработанный GitHub. Позволяет разрабатывать нативные графические приложения для настольных операционных систем с помощью веб-технологий. Фреймворк включает в себя Node.js для работы с back-end и библиотеку рендеринга из Chromium.

На базе Electron построен не только текстовый редактор для программистов Atom, но и такие программные продукты для разработчиков, как Visual Studio Code, Light Table (начиная с версии 0.8), Ionic Lab, Avocode, REPL-консоль Mancy для фреймворков Node.js и Meteor.js, Mongotron — GUI-менеджер для MongoDB. Кроме того, на основе этого фреймворка написано клиентское приложение чата Slack, Skype, Discord, десктопный клиент WordPress и многое другое [7].

### **3 Анализ средств разработки Back-End**

#### **Java Spring**

Spring Framework — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет большую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java. Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой Java Enterprise, несмотря на его масштабную интеграцию с ней, что является важной причиной его популярности.

Spring, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне

тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java [11].

## **Django**

Django — свободный фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC. Проект поддерживается организацией Django Software Foundation.

Сайт на Django строится из одного или нескольких приложений, которые рекомендуется делать отчуждаемыми и подключаемыми. Это одно из существенных архитектурных отличий этого фреймворка от некоторых других (например, Ruby on Rails). Один из основных принципов фреймворка — DRY (англ. Don't repeat yourself)

Также, в отличие от других фреймворков, обработчики URL в Django конфигурируются явно при помощи регулярных выражений.

Для работы с базой данных Django использует собственный ORM, в котором модель данных описывается классами Python, и по ней генерируется схема базы данных.

Архитектура Django похожа на «Модель-Представление-Контроллер» (MVC). Контроллер классической модели MVC примерно соответствует уровню, который в Django называется Представление (англ. View), а презентационная

логика Представления реализуется в Django уровнем Шаблонов (англ. Template). Из-за этого уровневую архитектуру Django часто называют «Модель-Шаблон-Представление» (MTV).

Первоначальная разработка Django как средства для работы новостных ресурсов достаточно сильно отразилась на его архитектуре: он предоставляет ряд средств, которые помогают в быстрой разработке веб-сайтов информационного характера. Так, например, разработчику не требуется создавать контроллеры и страницы для административной части сайта, в Django есть встроенное приложение для управления содержимым, которое можно включить в любой сайт, сделанный на Django, и которое может управлять сразу несколькими сайтами на одном сервере. Административное приложение позволяет создавать, изменять и удалять любые объекты наполнения сайта, протоколируя все совершённые действия, и предоставляет интерфейс для управления пользователями и группами (с пообъектным назначением прав).

В дистрибутив Django также включены приложения для системы комментариев, синдикации RSS и Atom, «статических страниц» (которыми можно управлять без необходимости писать контроллеры и представления), перенаправления URL и другое.

Основные возможности:

- ORM, API доступа к БД с поддержкой транзакций;
- встроенный интерфейс администратора, с уже имеющимися переводами на многие языки;
- диспетчер URL на основе регулярных выражений [14].

## Node

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из

узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [2].

## **PHP**

PHP — скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-сайтов.

В области веб-программирования, в частности серверной части, PHP — один из популярных сценарных языков (наряду с JSP, Perl и языками, используемыми в ASP.NET).

Популярность в области построения веб-сайтов определяется наличием большого набора встроенных средств и дополнительных модулей для разработки веб-приложений. Основные из них:

- автоматическое извлечение POST- и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
- взаимодействие с большим количеством различных систем управления базами данных через дополнительные модули (MySQL, MySQLi, SQLite, PostgreSQL, Oracle (OCI8), Oracle, Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus

Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird / InterBase, Paradox File Access, MaxDB, Интерфейс PDO), Redis;

- автоматизированная отправка HTTP-заголовков;
- работа с HTTP-авторизацией;
- работа с cookies и сессиями;
- работа с локальными и удалёнными файлами, сокетами;
- обработка файлов, загружаемых на сервер;
- работа с XForms [15].



## **4 Формирование требований**

Компания – сеть спортивных залов. Предоставляет услуги тренажерного зала и единоборств. В любом из филиалов клиент может приобрести абонемент и посещать занятия [1]. Система учета должна автоматизировать деятельность администратора спортивного центра. Система должна давать возможность манипулировать данными о клиентах, абонементах и тренерах.

Также сеть спортивных залов нуждается в хранении информации о посещаемости залов в целях анализа работы предприятия и эффективности тренеров. Система должна предоставлять отчет о посещаемости зала, а также отчет «начисления при продаже», используемый при выдаче заработной платы тренерам и анализе их эффективности.

Ниже представлены требования, сформированные, опираясь на текущую систему учета.

### **4.1 Требования к дизайну**

Приложение должно иметь преимущественно темные стили, чтобы обеспечить пользователю комфортную работу в ночное время суток. Так как приложение предназначено для чтения и редактирования данных, шрифты должны быть читаемыми и достаточно крупными. Окна должны отображать информацию в структурированном виде. Упор делается на восприятии информации, а не на внешнем виде окон.

### **4.2 Функциональные требования**

Определим основные функциональные требования:

- Авторизация - при запуске приложения пользователя встречает окно авторизации, требующее ввода имени пользователя и пароля.

- Отметка посещения - пользователь может отмечать посещения клиентами спортивного зала.
- Просмотр информации о клиентах - пользователь может посмотреть информацию о клиентах, выполнить поиск клиента по нескольким критериям, таким как ФИО и номер телефона.
- Добавление/редактирование/удаление клиента - пользователь может занести нового клиента в систему учета. Имеется возможность редактировать и удалять имеющихся клиентов.
- Добавление/редактирование/удаление абонемента - пользователь может занести новый абонемент в систему учета. Имеется возможность редактировать и удалять имеющиеся абонементы.
- Добавление/редактирование/удаление оплаты - пользователь может занести новую оплату в систему учета. Имеется возможность удалять имеющиеся оплаты.
- Добавление/редактирование/удаление тренера - пользователь может занести нового тренера в систему учета. Имеется возможность редактировать и удалять имеющихся тренеров.
- Добавление/редактирование/удаление вида абонемента - пользователь может занести новый вид абонемента в систему учета. Имеется возможность редактировать и удалять имеющиеся виды абонементов.
- Просмотр посещений - пользователь может посмотреть список посещений спортивного зала. Имеется возможность поиска посещений конкретных клиентов и фильтрации по датам.
- Просмотр отчета деятельности предприятия - пользователь может посмотреть отчет «Начисления при продаже», содержащий информацию о деятельности предприятия, используемый при выдаче заработной платы тренерам и анализе их эффективности.

### **4.3 Требования к хранению данных**

Вся информация, предоставляемая приложением, должна храниться в реляционной базе данных в структурированном виде в таблицах. Исключения составляют фотографии клиентов зала. В базе данных размещаются только ссылки на них.

### **4.4 Требования к ПО**

Для функционирования приложения на сервере должно присутствовать следующее программное обеспечение:

- Реляционная СУБД PostgreSQL версии 7.4 или выше;
- Node версии 8.8 или выше.

Для функционирования приложения на клиенте должно присутствовать следующее программное обеспечение:

- Node версии 8.8 или выше;
- браузер Google Chrome версии 57 и выше.

### **4.5 Требования к безопасности**

Так как клиент-серверное взаимодействие предполагает передачу персональных данных посетителей зала, возникает необходимо обеспечить их конфиденциальность. Необходимо использование протокола HTTPS для SSL/TLS-шифрования. Для защиты маршрутов необходима авторизация и выдача токенов. Все действия по записи в БД должны протоколироваться в отдельные таблицы. Ежедневно должна создаваться и загружаться в облако резервная копия БД.

## 5 Выбор средств реализации

### 5.1 Клиентская логика

Для реализации клиентской логики из возможных вариантов был выбран Electron по следующим причинам:

- наличие опыта в web-разработке. Это значит, что качество продукта останется на высоком уровне, а скорость разработки существенно возрастет, что позволит заказчику сэкономить средства;
- при разработке используется HTML и CSS, что при желании позволит создать гибкий и уникальный пользовательский интерфейс без использования стандартных заранее определенных компонент.

Таким образом, при реализации клиентской логики будет использоваться стек технологий, описанный ниже.

#### Node.js

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Преимущества:

- асинхронность в сочетании с событийным подходом;
- легкость и скорость написания;
- легковесность;
- встроенный менеджер пакетов и наличие дерева зависимостей;
- постоянное развитие [2].

## **Vue.js**

Vue.js - JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

Преимущества:

- простота;
- отсутствие требований к стеку;
- легковесность;
- высокая скорость разработки;
- хорошая документация.

Недостатки:

- Работа над состоянием приложения происходит “под капотом”;
- Компонентный подход во Vue не так гибок и очевиден;
- Отсутствие поддержки крупных проектов [6].

## **Electron**

Electron.js - фреймворк, разработанный GitHub. Позволяет разрабатывать нативные графические приложения для настольных операционных систем с помощью веб-технологий. Фреймворк включает в себя Node.js и библиотеку рендеринга из Chromium.

Преимущества:

- возможность работать вместе с Vue.js;
- кроссплатформенность;
- богатый набор как встроенных, так и сторонних компонентов;
- простота в использовании;
- хорошая документация.

Недостатки:

- большое потребление оперативной памяти [7].

## 5.2 Серверная логика

Для реализации серверной логики из возможных вариантов были выбраны Node.js и фреймворк Express.js по следующим причинам:

- форматом клиент-серверного обмена данными является JSON, что привычно для Vue.js;
- при разработке, как и на клиенте, используется язык программирования JavaScript, что упрощает написание кода.

Таким образом, при реализации серверной логики будет использоваться стек технологий, описанный ниже.

### Node.js

Node.js — программная платформа, основанная на движке V8 (транслирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода-вывода через

свой API (написанный на C++), подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера, но есть возможность разрабатывать на Node.js и десктопные оконные приложения. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом.

Преимущества:

- асинхронность в сочетании с событийным подходом;
- легкость и скорость написания;
- легковесность;
- встроенный менеджер пакетов и наличие дерева зависимостей;
- постоянное развитие [2].

## **Express.js**

Express.js - фреймворк web-приложений для Node.js, реализованный как свободное и открытое программное обеспечение под лицензией MIT. Он спроектирован для создания веб-приложений и API. Де-факто является стандартным каркасом для Node.js.

Преимущества:

- простота;
- гибкость;
- хорошая масштабируемость;
- развитое сообщество;
- подробная документация;
- широкий выбор подключаемых модулей;

## Недостатки

- большой объем ручной работы;
- используется устаревший подход *callbacks* функций [3].

## PostgreSQL

PostgreSQL — это популярная свободная объектно-реляционная система управления базами данных. PostgreSQL базируется на языке SQL и поддерживает многочисленные возможности. PostgreSQL предоставляет множество различных возможностей, достаточно надежна и имеет хорошие характеристики по производительности. Она работает практически на всех UNIX-платформах, включая UNIX-подобные системы

### Преимущества:

- поддержка БД неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
- наследование;
- легкая расширяемость;

### Недостатки:

- неэффективность архитектуры в плане выполнения записи;
- неэффективная репликация данных;
- случаи повреждения таблиц;
- проблемы с MVCC на репликах;
- трудности с обновлением [4].

## Knex.js



Knex.js – фреймворк для построения запросов с большим количеством диалектов (MSSQL, MySQL, PostgreSQL, SQLite3, Oracle), используемый node.js, поддерживающий транзакции, пулы соединений, потоковые запросы и промисы [5].

## **6 Проектирование**

### **6.1 Проектирование клиента**

Так как создание новой системы учета ведется на основе старой системы «USU», интерфейс пользователя будет перенесен с некоторыми поправками. Интерфейс будет представлять из себя приложение с горизонтальным меню сверху, которое обеспечивает навигацию для окон «Отметить», «Абонементы», «Клиенты», «Тренеры», «Оплаты», «Отчет» для работы с соответствующими сущностями из базы данных. Для работы с сущностью «Виды абонементов» будет создана кнопка с переходом в разделе «Абонементы». Для просмотра отчета «Начисления при продаже» будет создана кнопка с переходом в разделе «Отчет». Также в горизонтальном меню будет создана кнопка, по нажатии которой открывается меню настроек, позволяющее изменить пароль управляющего и указать стандартную ставку тренера.

На рисунке 12 представлена структура приложения, отражающая систему окон и переходов между окнами.

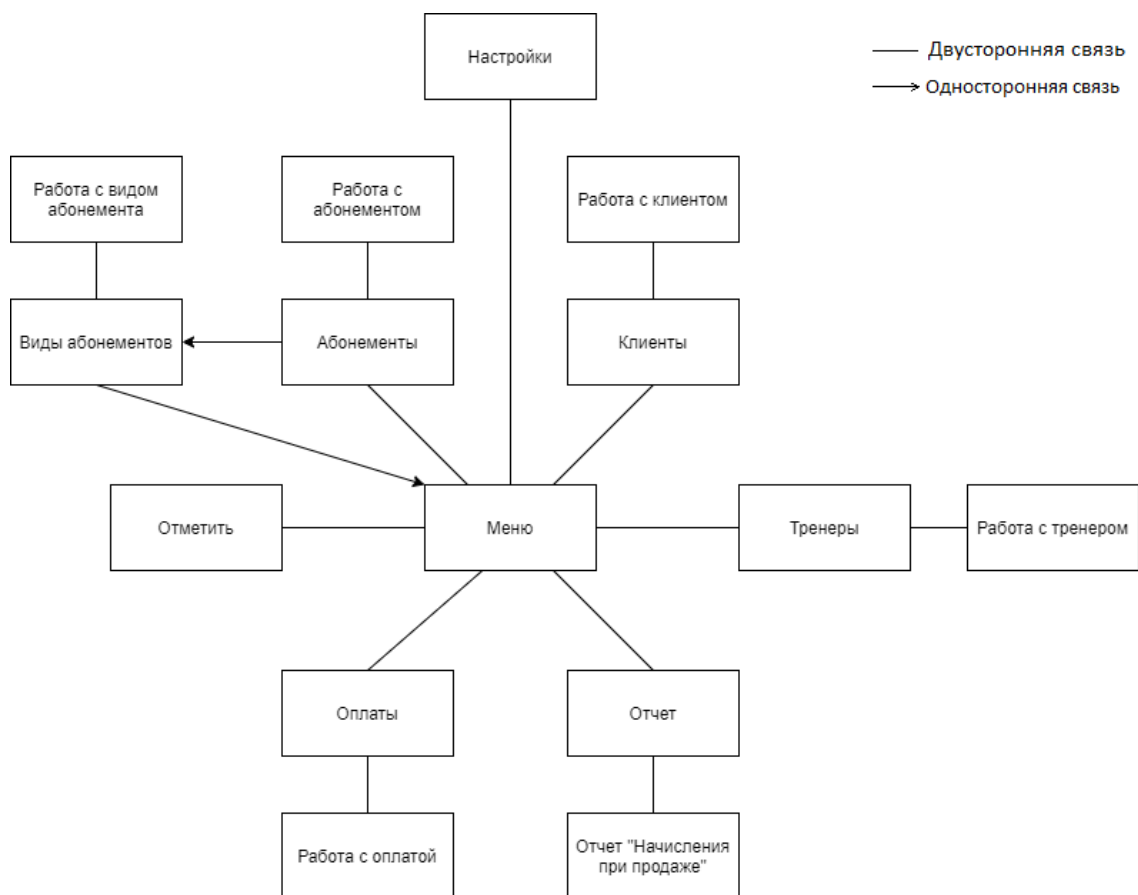


Рисунок 12 – Структура приложения

## 6.2 Проектирование сервера

### Проектирование базы данных

База данных должна содержать информацию о клиентах, тренерах, абонементов и оплатах. Также, в дополнение к основной информации, база данных должна хранить информацию о посещаемости зала.

В соответствии с предметной областью система строится с учётом следующих особенностей:

- каждый клиент может иметь несколько абонементов;
- каждый абонемент привязан только к одному виду абонемента;
- каждый абонемент привязан только к одному клиенту;

- каждый абонемент привязан только к одному тренеру;
- каждый тренер может вести занятия у нескольких клиентов.

Выделим базовые сущности этой предметной области:

- клиенты;
- тренеры;
- виды абонементов;
- абонементы;
- оплаты;
- посещения.

Описание сущностей и отношений представлено на рисунке 13.

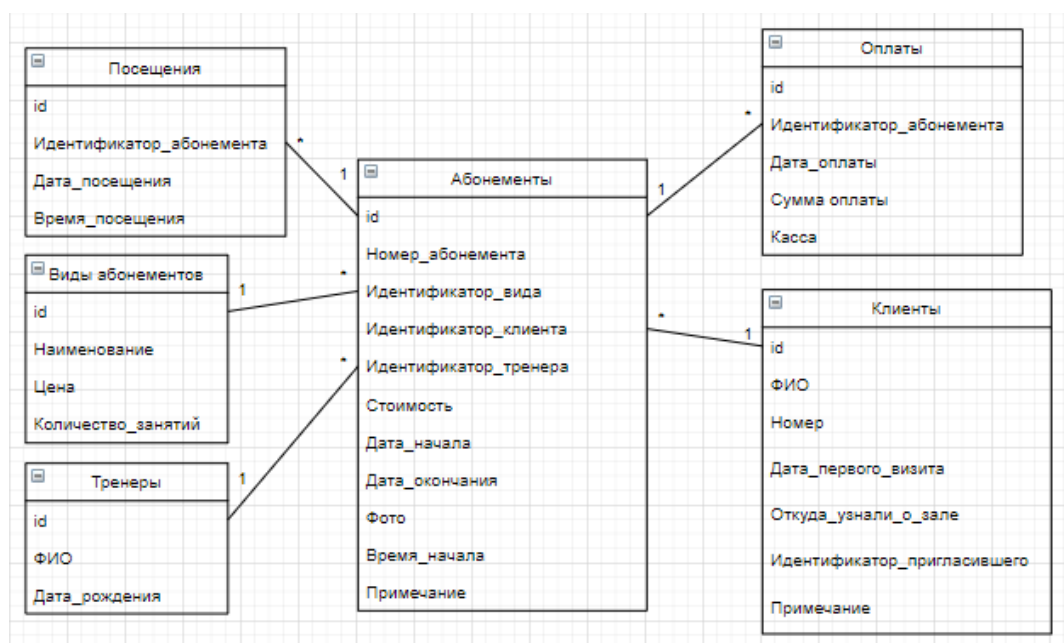


Рисунок 13 – Диаграмма сущностей и связей

## Составление реляционных отношений

Схемы отношений для определенных ранее сущностей представлены в таблицах 1-6.

Таблица 1 – Схема отношения «Абонементы»

Содержание	Имя	Тип	Примечания
Идентификатор	id	SERIAL	Первичный ключ
Номер абонемента	sub_number	VARCHAR(13)	Обязательное поле, уникальное поле
Идентификатор вида	type_id	INTEGER	
Идентификатор клиента	client_id	INTEGER	Обязательное поле
Идентификатор тренера	trainer_id	INTEGER	
Осталось оплатить	left_to_pay	INTEGER	Обязательное поле
Дата начала	begin_date	DATE	Обязательное поле
Дата окончания	end_date	DATE	Обязательное поле
Осталось занятий	training_left	SMALLINT	Обязательное поле
Время начала	start_time	TIME	Обязательное поле
Примечание	note	VARCHAR(200)	

Таблица 2 – Схема отношения «Виды абонементов»

Содержание	Имя	Тип	Примечания
Идентификатор	id	SERIAL	Первичный ключ
Название	title	VARCHAR(50)	Обязательное поле
Цена	cost	INTEGER	Обязательное поле
Количество занятий	training	SMALLINT	Обязательное поле

Таблица 3 – Схема отношения «Клиенты»

Содержание	Имя	Тип	Примечания
Идентификатор	id	SERIAL	Первичный ключ
ФИО	fio	VARCHAR(100)	Обязательное поле
Номер телефона	phone_number	VARCHAR(18)	Обязательное поле
Дата первого визита	first_visit_date	DATE	
Откуда узнали о зале	how_to_find	VARCHAR(16)	
Идентификатор пригласившего	inviter_id	INTEGER	
Примечание	note	VARCHAR(200)	

Таблица 4 – Схема отношения «Оплаты»

Содержание	Имя	Тип	Примечания
Идентификатор	id	SERIAL	Первичный ключ
Номер абонемента	sub_id	INTEGER	Обязательное поле
Дата оплаты	payment_date	DATE	Обязательное поле
Сумма оплаты	payment_amount	INTEGER	Обязательное поле
Метод оплаты	payment_method	VARCHAR(3)	Обязательное поле
Ставка тренера	interest_rate	INTEGER	Обязательное поле

Таблица 5 – Схема отношения «Тренеры»

Содержание	Имя	Тип	Примечания
Идентификатор	id	SERIAL	Первичный ключ
ФИО	fio	VARCHAR(100)	Обязательное поле
Дата рождения	date_birth	DATE	

Таблица 6 – Схема отношения «Посещения»

Содержание	Имя	Тип	Примечания
Идентификатор	id	SERIAL	Первичный ключ
Идентификатор абонента	sub_id	INTEGER	Обязательное поле
Дата визита	visit_date	DATE	Обязательное поле
Время визита	visit_time	TIME	Обязательное поле

### Нормализация полученных отношений до 4НФ

Отношения находятся в 1НФ, так как все атрибуты простые, все домены содержат скалярные значения, строки не повторяются, так как в каждой таблице используется первичный ключ с автоинкрементом.

Отношения находятся во 2НФ, так как они находятся в 1НФ, каждый не ключевой атрибут неприводимо зависит от первичного ключа.

Отношения находятся в 3НФ, так как они находятся во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Все отношения имеют только один первичный ключ, поэтому находятся в НФБК.

Отношения находятся в 4НФ, так как отсутствуют многозначные зависимости.

### Определение дополнительных ограничений целостности

Перечислим дополнительные ограничения целостности:

- идентификатор типа не может быть меньше нуля;
- идентификатор клиента не может быть меньше нуля;
- идентификатор тренера не может быть меньше нуля;

- дата начала не может быть больше даты окончания;
- при удалении клиента удаляются все его абонементы (внешний ключ);
- при удалении тренера, все абонементы, привязанные к нему, меняют значение идентификатора тренера на NULL (внешний ключ);
- при удалении вида абонемента, все абонементы, привязанные к нему, меняют значение идентификатора вида абонемента на NULL (внешний ключ);
- при удалении вида абонемента, все абонементы, привязанные к нему, меняют значение идентификатора вида абонемента на NULL (внешний ключ).

Список запросов, используемых при создании БД описан в приложении А.

## Проектирование API

Формат обмена данными - JSON (JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript. Произошел из стандарта ECMA-262, формат считается независимым от JavaScript и может использоваться практически с любым языком программирования.

Описание API сервера представлено в таблице 7.

Таблица 7 - Описание API сервера

URL	Тип	Значение	Тело запроса	Тело ответа
v1/visits/getLatest	POST	Получить список последних посещений	beg_range, end_range	id, fio, sub_number, visit_date, visit_time
v1/visits/getByFio	POST	Получить список посещений по ФИО	beg_range, end_range, fio	id, fio, sub_number, visit_date, visit_time



Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/visits/getBySubNumber	POST	Получить список посещений по номеру абонента	beg_range, end_range, sub_number	id, fio, sub_number, visit_date, visit_time
v1/types/getLatest	GET	Получить список видов абонентов		id, title, cost, training
v1/types/edit		Изменить вид абонента	id, title, cost, training	
v1/types/findByTitle	POST	Получить виды абонента по наименованию	title	id, title, cost, training
v1/types/add	POST	Добавить вид абонента	title, cost, training	
v1/trainers/getLatest	GET	Получить список последних тренеров		id, fio, date_birth
v1/trainers/edit	POST	Изменить тренера	fio, date_birth	

Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/trainers/findByFio	POST	Получить список тренеров по ФИО	fio	id, fio, date_birth
v1/trainers/add	POST	Добавить тренера	fio, date_birth	
v1/trainers/remove	GET	Удалить тренера		
v1/subs/extend	POST	Продлить абонемент	id, beg_date, end_date	
v1/subs/checkUniq	POST	Проверить уникальность номера абонемента	sub_number	uniqAns
v1/subs/getLatest	POST	Получить список последних абонементов		sub_id, sub_number, start_time, note, begin_date, end_date, training_left, left_to_pay, client_fio, client_id, phone_number, title, training, cost, type_id, trainer_id, trainer_fio

Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/subs/getByFio	POST	Получить список абонементов по ФИО клиента	fio	sub_id, sub_number, start_time, note, begin_date, end_date, training_left, left_to_pay, client_fio, client_id, phone_number, title, training, cost, type_id, trainer_id, trainer_fio
v1/subs/getByPhoneNumber	POST	Получить список абонементов по номеру телефона	phone_number	sub_id, sub_number, start_time, note, begin_date, end_date, training_left, left_to_pay, client_fio, client_id, phone_number, title, training, cost, type_id, trainer_id, trainer_fio

Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/subs/getBySubNumber	POST	Получить список абонементов по номеру абонемента	sub_number	sub_id, sub_number, start_time, note, begin_date, end_date, training_left, left_to_pay, client_fio, client_id, phone_number, title, training, cost, type_id, trainer_id, trainer_fio
v1/subs/edit	POST	Изменить абонемент	sub_id, sub_number, start_time, note, begin_date, end_date, training_left, left_to_pay, client_fio, client_id, phone_number, title, training, cost, type_id, trainer_id, trainer_fio	

Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/subs/add	POST	Добавить абонемент	sub_id, sub_number, start_time, note, begin_date, end_date, training_left, left_to_pay, client_fio, client_id, phone_number, title, training, cost, type_id, trainer_id, trainer_fio	
v1/subs/remove	GET	Удалить абонемент	id	
v1/clients/getLatest	GET	Получить список последних клиентов		id, fio, phone_number, first_visit_date, how_to_find, inviter_id
v1/clients/edit	POST	Изменить клиента	id, fio, phone_number, first_visit_date, how_to_find, inviter_id, note, photo,	

Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/clients/getByPhoneNumber	POST	Получить список клиентов по номеру телефона	phone_number	id, fio, phone_number, first_visit_date, how_to_find, inviter_id, note, photo,
v1/clients/getByFio	POST	Получить список клиентов по ФИО	fio	id, fio, phone_number, first_visit_date, how_to_find, inviter_id, note, photo,
v1/clients/add	POST	Добавить клиента	id, fio, phone_number, first_visit_date, how_to_find, inviter_id, note, photo	
v1/clients/remove	GET	Удалить клиента	id	
v1/auth/login	POST	Авторизация	username, password	accessToken

Продолжение таблицы 7

URL	Тип	Значение	Тело запроса	Тело ответа
v1/mark/getInfo	POST	Получить данные абонемента	sub_number	sub_id, begin_date, note, start_time, end_date, training_left, left_to_pay, fio, title, training, cost, id,
v1/mark/markVisit	POST	Отметить посещение	id	
v1/report/getTrainerList	GET	Получить список тренеров		id, fio
v1/report/getReport	POST	Получить отчет	id, time, beg, end,	payment_date interest_rate payment_amount sub_number start_time fio title cost training

## 7 Реализация

### 7.1 Реализация сервера

#### Описание структуры проекта

Инициализируем express приложение со структурой проекта, представленной на рисунке 14.

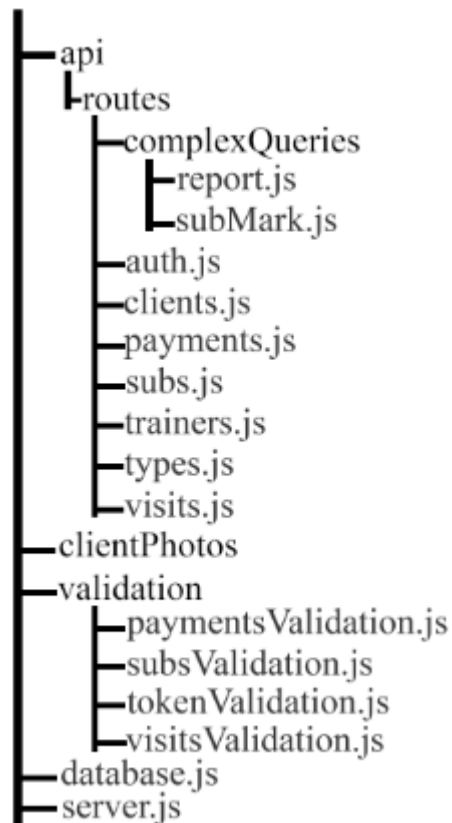


Рисунок 14 - Структура проекта

Назначение каталогов и файлов:

- /api/routes/ –маршруты;
- /clientPhotos/ –хранение фотографий клиентов;
- /validation/ –валидация, используемая в некоторых маршрутах;
- /database.js – подключение к базе данных;
- /server.js – описание маршрутов и запуск сервера.



## Описание маршрутов и запуск сервера

Для взаимодействия клиента с данными спортивного центра сервер будет использовать маршруты. Для каждой сущности из БД будет создан свой маршрут.

Ниже описан процесс создания маршрутов

```
const typeRoutes = require("./api/routes/types")
const trainersRoutes = require("./api/routes/trainers")
const clientsRoutes = require("./api/routes/clients")
const subsRoutes = require("./api/routes/subs")
const paymentsRoutes = require("./api/routes/payments")
const visitsRoutes = require("./api/routes/visits")
const subMark = require("./api/routes/complexQueries/subMark")
const report = require("./api/routes/complexQueries/report")
const auth = require("./api/routes/auth")

app.use("/v1/types", typeRoutes)
app.use("/v1/trainers", trainersRoutes)
app.use("/v1/clients", clientsRoutes)
app.use("/v1/subs", subsRoutes)
app.use("/v1/payments", paymentsRoutes)
app.use("/v1/visits", visitsRoutes)
app.use("/v1/report", report)
app.use("/v1/mark", subMark)
app.use("/v1/auth", auth)
```

Так как клиент-серверное взаимодействие предполагает передачу персональных данных, возникает необходимо обеспечить их конфиденциальность. Для этого будет использоваться протокол HTTPS. Самоподписанный сертификат и ключ были созданы при помощи OpenSSL.

```
https.createServer({
  key: fs.readFileSync('server_key.pem'),
  cert: fs.readFileSync('server.crt.pem')
}, app)
.listen(3000, function () {
  console.log('start')
})
```

## Авторизация и внедрение JSON Web Token

Для защиты маршрутов будет использоваться авторизация. Авторизованному пользователю выдается JSON Web Token (JWT). При обращении к серверу, авторизованный клиент будет использовать полученный

токен для получения доступа к любым данным. Для хранения информации о пользователях, в БД будет добавлена таблица users, хранящая пары имя пользователя и пароль. Ниже описан процесс авторизации и выдачи JWT.

```
const accessTokenSecret = 'Qm7UNBio86hJ79Bz000';

router.post("/login", async (req, res) => {
  let user = await database
    .select('username', 'userpass')
    .from('users')
    .where('username', req.body.username)
    .first()

  if(!user) {
    res.send('not exists')
  }
  if(user.userpass !== req.body.password) {
    res.send('not exists')
  }

  const accessToken = jwt.sign({ username: user.username }, accessTokenSecret, {
    expiresIn: '17h' });
  res.send(accessToken)
})
```

Проверка токена производится при вызове любого маршрута кроме авторизации. Ниже представлена реализация проверки.

```
const accessTokenSecret = 'Qm7UNBio86hJ79Bz000';

module.exports = (req, res, next) => {
  const authHeader = req.headers.authorization

  if (authHeader) {
    const token = authHeader.split(' ')[1];

    jwt.verify(token, accessTokenSecret, (err, user) => {
      if (err) {
        return res.sendStatus(403);
      }
      next();
    });
  } else {
    res.sendStatus(401);
  }
};
```

## Резервное копирование базы данных

Для предотвращения утери базы данных при возникновении каких-либо неисправностей был реализован скрипт, сохраняющий дамп базы в облако. Ниже приведен текст скрипта на win10 powershell.

```
Set-Location "C:\Program Files\PostgreSQL\12\bin"
```

```
.\pg_dump.exe --dbname=postgresql://postgres:12345@localhost:5432/sport > sportdump.sql
```

Скрипт, запускаемый ежедневно в 20:00 с помощью планировщика задач win10 представлен на рисунке 15.

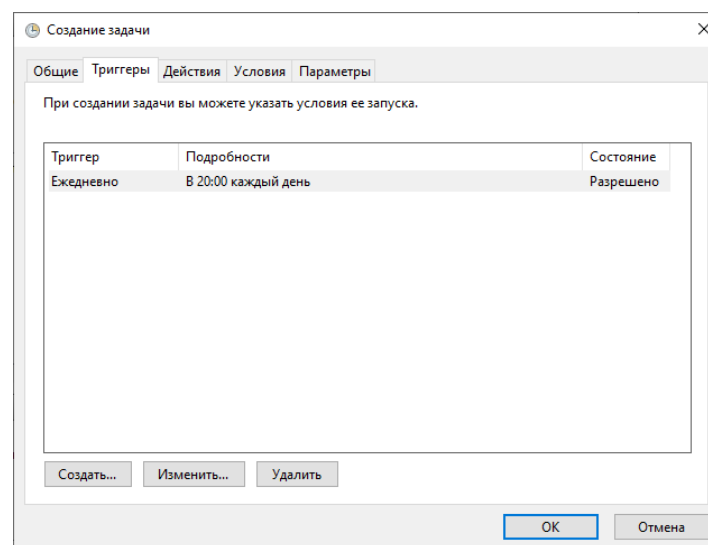


Рисунок 15 – Добавление скрипта резервного копирования БД в планировщик задач

## Описание реализации маршрутов

Рассмотрим реализацию маршрутов на примере маршрута *subs*, обеспечивающего взаимодействие клиента с сущностью *Абонементы*.

Для работы с БД используется *knex.js*. Для обращения к базе данных был создан скрипт *database.js*. Его описание представлено ниже.

```
const knex = require("knex")
const database = knex({
```

```

    client: "pg",
    connection: {
      host: "127.0.0.1",
      user: "postgres",
      password: "12345",
      database: "sport"
    }
  });

module.exports = database

```

Ниже представлен код файла *subs.js*, реализующего маршрут *subs*.

```

// Продление абонеента
router.post(«/extend», authJwt, async(req, res) => {
  // Получение идентификатора вида абонеента
  let typeQuery = await database
    .select('type_id', 'left_to_pay')
    .from('subs')
    .where({
      id: req.body.id
    })
    .first()
  // Получение информации о виде абонеента для обновления полей оставшихся
занятий и оплат
  let typeAdditionalInfo = await database
    .select()
    .from('types')
    .where({
      id: typeQuery.type_id
    })
    .first()

  // Запись новых дат действия абонеента и обновление оплат и занятий
  await database('subs').
  Update({
    begin_date: req.body.beg_date,
    end_date: req.body.end_date,
    left_to_pay: typeAdditionalInfo.cost + typeQuery.left_to_pay,
    training_left: typeAdditionalInfo.training
  })
  .where({
    id: req.body.id
  })

  res.sendStatus(200)
})

// Проверка уникальности номера абонеента для добавления абонеента
router.post(«/checkUniq», authJwt, async(req, res) => {
  let field = await database('subs')
    .select()
    .where({
      sub_number: req.body.sub_number
    })

```

```

        .first()

        var uniqAns = field === undefined ? true : false
        res.send(uniqAns)
    })

    // Получить список последних редактированных абонементов
    router.get("/getLatest", authJwt, async (req, res) => {
        let query = await database
            .select('subs.id as sub_id', 'sub_number', 'start_time', 'subs.note',
                'begin_date', 'end_date', 'training_left', 'left_to_pay', 'clients.fio as client_fio',
                'clients.id as client_id', 'phone_number', 'title', 'training', 'cost', 'types.id as
                type_id', 'trainers.id as trainer_id', 'trainers.fio as trainer_fio')
            .from('subs')
            .leftJoin('clients', 'subs.client_id', 'clients.id')
            .leftJoin('types', 'subs.type_id', 'types.id')
            .leftJoin('trainers', 'subs.trainer_id', 'trainers.id')
            .limit(40)

        res.send(query)
    })

    // Поиск абонемента по ФИО
    router.post("/getSubByFio", authJwt, async (req, res) => {
        let query = await database
            .select('subs.id as sub_id', 'sub_number', 'start_time', 'subs.note',
                'begin_date', 'end_date', 'training_left', 'left_to_pay', 'clients.fio as client_fio',
                'clients.id as client_id', 'phone_number', 'title', 'training', 'cost', 'types.id as
                type_id', 'trainers.id as trainer_id', 'trainers.fio as trainer_fio')
            .from('subs')
            .leftJoin('clients', 'subs.client_id', 'clients.id')
            .leftJoin('types', 'subs.type_id', 'types.id')
            .leftJoin('trainers', 'subs.trainer_id', 'trainers.id')
            .where('clients.fio', 'ilike' , `%${req.body.fio}%`)
            .limit(20)

        res.send(query);
    })

    // Добавление абонемента
    router.post("/add", authJwt, async (req, res) => {
        // Проверка ввода вида абонемента, тренера и клиента на существование
        let validatedIDs = await valid({
            type_id: req.body.type_id,
            trainer_id: req.body.trainer_id,
            client_id: req.body.client_id
        });

        if(validatedIDs.type_id === false || validatedIDs.trainer_id === false ||
            validatedIDs.client_id === false) {
            res.sendStatus(400);
            return;
        }

        // Получение информации о виде абонемента
        let typeAdditionalInfo = await database
            .select()
            .from('types')

```

```

.where({
  id: req.body.type_id
});

// Добавление записи
await database('subs').
insert({
  sub_number: req.body.sub_number,
  sub_status: 1,
  type_id: req.body.type_id,
  client_id: req.body.client_id,
  trainer_id: req.body.trainer_id,
  left_to_pay: typeAdditionalInfo[0].cost,
  begin_date: req.body.begin_date,
  end_date: req.body.end_date,
  training_left: typeAdditionalInfo[0].training,
  start_time: req.body.start_time,
  note: req.body.note
});

res.sendStatus(200);
});

```

## 7.2 Реализация клиента

### Описание структуры проекта

Для создания системы учета будет использоваться готовый шаблон проекта *electron-vue*, позволяющий работать Vue.js вместе с Electron.js. Структура проекта представлена на рисунке 16 [8].

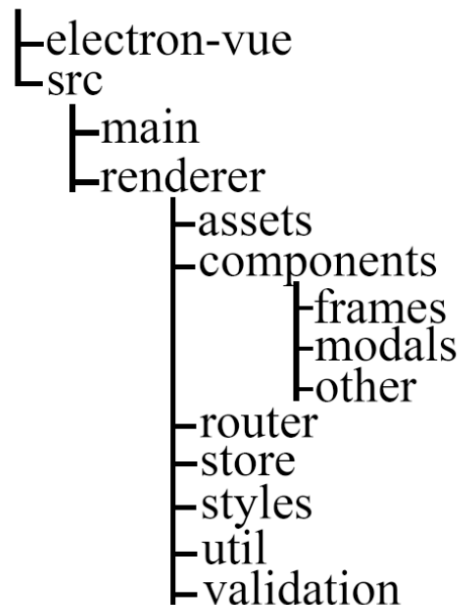


Рисунок 16 - Структура проекта

Назначение каталогов и файлов:

- /electron-vue/ – совместная работа Electron.js и Vue.js;
- /src/main/ – конфигурация и запуск приложения ;
- /src/renderer/assets/ – изображения;
- /src/renderer/components/frames/ – компоненты основных окон;
- /src/renderer/components/modals/ – компоненты диалоговых окон;
- /src/renderer/components/other/ – вспомогательные компоненты;
- /src/renderer/router/ – маршруты;
- /src/renderer/store/ – Vuex хранилище;
- /src/renderer/styles/ – стили;
- /src/renderer/util/ – вспомогательные скрипты;
- /src/renderer/validation/ – валидация.

## Создание Vuex хранилища

Для глобального хранения переменных во Vue.js предусмотрено хранилище Vuex. Хранилище будет использоваться для хранения токена доступа авторизованного пользователя и некоторых состояний приложения для упрощения реализации сложной логики при взаимодействии большого количества вложенных окон.

Хранилище делится на несколько модулей. Ниже приведено описание всех модулей.

Модуль *subsModule* предназначен для хранения состояний окна «Абонементы» и его дочерних окон. Состояние *isAddOperation* показывает, находится ли окно в состоянии добавления абонемента. Состояние *isEditOperation* показывает, находится ли окно в состоянии изменения абонемента.

```
const subsModule = {
```

```

namespaced: true,
state: {
  isAddOperation: false,
  isEditOperation: false
},
getters: {
  isAddOperation: state => {
    return state.isAddOperation
  },
  isEditOperation: state => {
    return state.isEditOperation
  }
},
mutations: {
  setIsAddOperation (state, value) {
    state.isAddOperation = value
  },
  setIsEditOperation (state, value) {
    state.isEditOperation = value
  }
}
}

```

Модуль *clientsModule* отвечает за хранения состояния окна «Клиенты». *isPictureTaken* – был ли сделан снимок. *isAddOperation* – находится ли окно в состоянии добавления клиента. *isEditOperation* – находится ли окно в состоянии редактирования клиента. *pictureFromDatabase* – изображение клиента, полученное из БД.

```

const clientsModule = {
  namespaced: true,
  state: {
    isAddOperation: false,
    isPictureTaken: false,
    isEditOperation: false,
    isVideoShow: false,
    pictureFromDatabase: '',
    clientPhoto: ''
  },
  getters: {
    pictureFromDatabase: state => {
      return state.pictureFromDatabase
    },
    clientPhoto: state => {
      return state.clientPhoto
    },
    isAddOperation: state => {
      return state.isAddOperation
    },
    isPictureTaken: state => {
      return state.isPictureTaken
    },
    isEditOperation: state => {
      return state.isEditOperation
    }
  }
}

```



```

    },
    isVideoShow: state => {
      return !state.isPictureTaken
    }
  },
  mutations: {
    setClientPhoto (state, value) {
      state.clientPhoto = value
    },
    setPictureFromDatabase (state, value) {
      state.pictureFromDatabase = value
    },
    setIsAddOperation (state, value) {
      state.isAddOperation = value
    },
    setIsEditOperation (state, value) {
      state.isEditOperation = value
    },
    setIsPictureTaken (state, value) {
      state.isPictureTaken = value
    }
  }
}

```

Для объединения модулей создается экземпляр `Vuex.Store`, который также содержит свойство `token`, хранящее токен доступа авторизованного пользователя.

```

export const store = new Vuex.Store({
  modules: {
    clientsFrame: clientsModule,
    subsFrame: subsModule
  },
  state: {
    token: ''
  },
  getters: {
    getToken: state => {
      return state.token
    }
  },
  mutations: {
    setToken (state, value) {
      state.token = value
    }
  }
})

```

## Реализация окна «Авторизация»

Для защиты маршрутов предусмотрено окно авторизации. Для того, чтобы получать данные с сервера, пользователь должен авторизоваться и получить токен доступа.

Ниже описан процесс реализации окна авторизации.

Основной блок – `auth-frame`. Занимает все пространство окна, включая меню. Отвечает за затемнение заднего фона. Пока данное окно не будет закрыто, пользователь не сможет работать с приложением. В основной блок вложен блок `auth-wrapper`, отвечающий за размер окна. Включает в себя две формы для ввода логина и пароля и кнопку «Вход». Окно авторизации представлено на рисунке 17.

```
<template>
  <div class="auth-frame">
    <div class="auth-wrapper">
      <div class="auth-title">Авторизуйтесь, чтобы начать работу</div>
      <div class="auth-body">
        <div class="auth-block">
          <div class="input-label">Имя пользователя</div>
          <div class="input-label">Пароль</div>
        </div>
        <div class="auth-block">
          <input class="user-input" v-model="username">
          <input type="password" class="user-input" v-model="password">
        </div>
      </div>
      <div class="button-wrapper" @click="accept">
        <button class="edit-type-button">Войти</button>
      </div>
    </div>
  </div>
</template>
```

Для проверки авторизации используется метод *accept*. Проводит валидацию введенного имени пользователя и пароля, и при успехе отправляет логин и пароль на сервер для проверки наличия такого пользователя в БД. Если пользователь существует, токен доступа записывается в Vuex хранилище.

```
async accept() {
  let sendObj = {
    username: this.username,
    password: this.password
  }

  let { isCorrect, alertMessage } = validate(sendObj)
  if(!isCorrect) {
    alert(alertMessage)
  }

  let res = await this.$axios.post('https://localhost:3000/v1/auth/login',
sendObj)
```

```

    if(res.data === 'not exists') {
      alert('Неверное имя пользователя или пароль')
      return
    }
    this.$store.commit('setToken', res.data)

    this.$emit('authDone')
  }
}

```

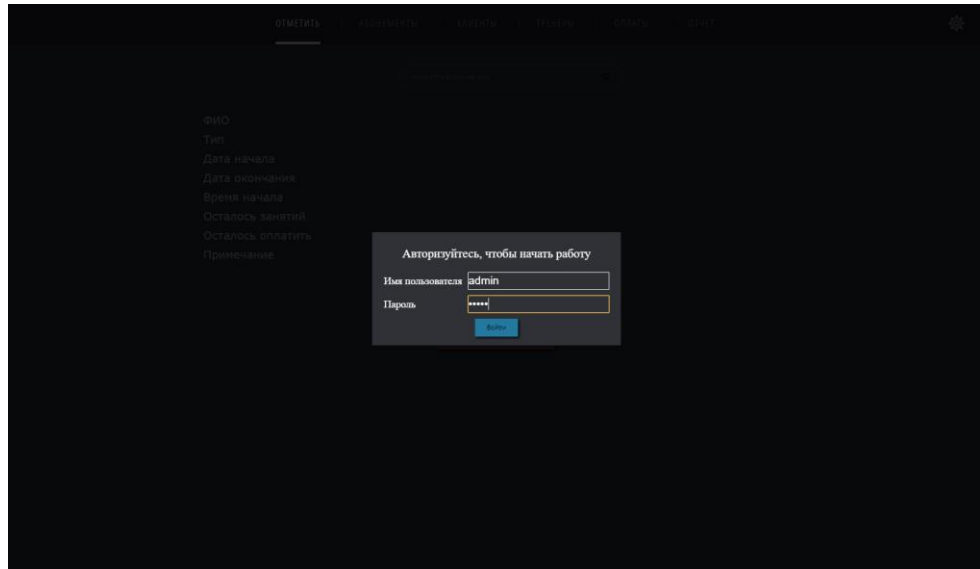


Рисунок 17 – Авторизация

## Реализация компоненты «Меню»

Ниже представлена реализация компоненты «Меню». Меню представлено на рисунке 18.

```

<template>
  <div>
    <ul class="menu-main">
      <li v-for="(route, idx) in routes" :key="route"
        @click="currentRoute=route">
        <router-link :to="'/' + route" :class="{ 'current': currentRoute ===
route }">
          {{ routeNames[idx] }}
        </router-link>
      </li>
    </ul>
    <div class="settings" @click="settingsClicked">
      
    </div>
    <settings
      v-show="settingsVisible"
      @settingsClose="settingsVisible = false"
      :key="settingsKey"
    />
  </div>

```

```

    </div>
</template>

<script>
import settings from './Settings'

export default {
  components: {
    settings
  },
  data() {
    return {
      settingsVisible: false,
      settingsKey: 0,
      currentRoute: 'mark',
      routes: [
        'mark',
        'subs',
        'clients',
        'trainers',
        'payments',
        'report'
      ],
      routeNames: [
        'Отметить',
        'Абонементы',
        'Клиенты',
        'Тренеры',
        'Оплаты',
        'Отчет'
      ]
    }
  }
}
</script>

<style scoped>
@import '../..//styles/menu.scss';
</style>

```



Рисунок 18 - Меню

## Реализация компоненты «Поисковая панель»

Компонента, имеющая поле для ввода текста и выпадающее меню для выбора критерия поиска, представлена на рисунке 19. Позволяет принимать на ввод данные пользователя и при изменении поля для ввода текста вызывает метод `search`, на вход которого передает введенный текст и критерий для поиска. Ниже представлено описание.

```

<template>
  <div class="search-frame">
    <div class="search-container">
      <input
        class="search-input"
        type="text"
        :placeholder="searchCriterion"
        v-model="userInput"
        @input="$emit('search', searchCriterion, userInput)"
      >
    </div>

    <select
      class="criterion-select"
      v-model="searchCriterion"
      @change="$emit('search', searchCriterion, userInput)"
    >
      <option v-for="item in options" :key="item">
        {{ item }}
      </option>
    </select>
  </div>
</template>

<script>
export default {
  props: {
    options: Array
  },
  data() {
    return {
      searchCriterion: this.options[0],
      userInput: ''
    }
  }
}
</script>

<style lang="scss" scoped>
@import '../styles/searchPanel.scss';
.search-frame {
  display: flex;
  justify-content: center;
  padding: 10px 0px 10px 0px;
}
</style>

```

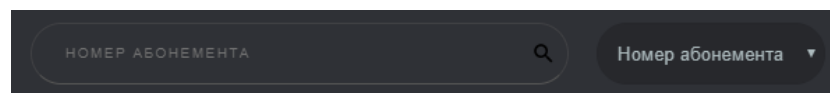


Рисунок 19 – Поисковая панель

**Реализация окон «Абонементы» и «Просмотр абонемента»**

Рассмотрим реализацию окон на примере окна «Абонементы». Окно содержит компоненту для поиска абонемента по нескольким критериям (номер абонемента, ФИО, номер телефона). Под навигационной панелью расположена таблица, хранящая результат поиска. По нажатии на строку в таблице открывается диалоговое окно «Просмотр абонемента», отображающее полную информацию о данном абонемента. Под таблицей находится кнопка для создания абонемента, а также кнопка для перехода в раздел «Виды абонементов». Внешний вид окна показан на рисунке 20.

В случае, если администратор захочет посмотреть информацию об абонемента, диалоговое окно представляет собой прямоугольную область, содержащую информацию об абонемента, а также кнопки «Изменить/Применить», «Удалить», «Продлить» и «X» (закрыть).

В случае, если администратор захочет добавить абонемент, диалоговое окно представляет собой прямоугольную область, содержащую поля для ввода информации об абонемента, а также кнопки «Добавить» и «X» (закрыть).

Подобная структура будет использоваться во всех окнах.

Ниже представлено описание окна «Абонементы» с соответствующими пояснениями.

Окно состоит из нескольких вложенных блоков (div). Первый блок *main-frame* занимает все пространство окна, кроме меню. Внутри него находятся блоки, содержащие поисковую панель, таблицу результатов и кнопки для добавления абонемента и перехода в раздел «Виды абонементов». Также внутри него находится компонента модального окна просмотра абонемента, которая не показывается пользователю до тех пор, пока он не начнет просмотр абонемента.

```
<template>
  <div class="main-frame">
    <search-panel
      v-bind:options="gridColumnsToShow"
```

```

        @search="setUserInput"
    />
    <div class="search-result-frame">
        <div class="search-result-header">
            <div
                v-for="item in gridColumnToShow"
                :key="item"
            >
                {{ item }}
            </div>
        </div>
        <div
            class="search-result"
            v-for="(node, idx) in subListToShow"
            :key="idx"
            @click="onRowClicked(idx)"
        >
            <div
                v-for="value in node"
                :key="value"
            >
                {{ value }}
            </div>
        </div>
    </div>
    <div class="add-sub-wrapper">
        <button class="add-type-button" @click="addSub">Добавить абонемент</button>
    </div>
    <div class="goto-sub-types-wrapper">
        <router-link to="/subTypes">
            <button class="goto-types-type-button">В раздел "Виды
абонементов"</button>
        </router-link>
    </div>
    <subModal
        v-bind:gridRows="gridColumns"
        v-bind:gridNodes="modalInfo"
        v-bind:receivedSubNumber="receivedSubNumber"
        v-show="modalShow"
        @modalClose="modalClose"
        @modalCloseX="modalShow = false"
    />
</div>
</template>

```

Далее в скрипте импортируются все нужные компоненты (поисковая панель, диалоговое окно просмотра абонемента, хранилище) и скрипты (валидация, приведение слов в нужный падеж, преобразование дат).

```

import searchPanel from '../other/searchPanel'
import subModal from '../modals/subsModal'
import { mapGetters } from 'vuex'

import trainCase from '../../util/trainingCase'
import convert from '../../util/dateConvert'

```

Для хранения переменных во Vue.js предусмотрен метод *data*. Область видимости – текущая компонента. Для того, чтобы пользователь мог просматривать список абонементов, были созданы объекты *gridColumnsToShow* и *subList*, хранящие наименования столбцов таблицы и список абонементов соответственно.

```
data() {  
  return {  
    searchCriterion: '',  
    gridColumns: [  
      "id",  
      "Номер абонемента",  
      "Клиент",  
      "Вид абонемента",  
      "Тренер",  
      "Время начала",  
      "Осталось занятий",  
      "Осталось оплатить",  
      "Примечание",  
      "Дата начала",  
      "Дата окончания",  
      "Дата"  
    ],  
    gridColumnsToShow: [  
      "Номер абонемента",  
      "ФИО",  
      "Номер телефона",  
    ],  
    subList: [],  
    modalShow: false,  
    modalInfo: {},  
    receivedSubNumber: '',  
    userInput: '',  
  }  
},
```

Для получения пользователем списка абонементов при открытии страницы был использован хук жизненного цикла *created*, который запускает метод *search* после создания окна.

```
created() {  
  this.search()  
},
```

Для поддержания реактивности во Vue.js используется свойство *computed* экземпляра *Vue*, позволяющее обновлять переменные, когда это необходимо. В нашем случае свойство используется для отображения списка абонементов в таблице. Список обновляется при изменении переменной *subList*, что позволяет отображать полученные с сервера данные без перезагрузки страницы.

```
computed: {  
  ...mapGetters({  
    isAddOperation: 'subsFrame/isAddOperation'  
  }),  
  subListToShow: function() {
```



```

        var newList = []
        this.subList.forEach(node => {
            newList.push({
                subNumber: node.subNumber,
                fio: node.fio,
                phoneNum: node.phoneNum
            })
        });

        return newList
    }
},

```

Для взаимодействия окна с компонентой поиска используется метод *setUserInput*, вызываемый компонентой поиска. Метод позволяет сохранять пользовательский ввод и выполнять поиск.

```

setUserInput(searchCriterion, userInput) {
    this.userInput = userInput
    this.searchCriterion = searchCriterion
    this.search()
},

```

Метод *addSub* делает диалоговое окно просмотра абонента видимым. Передает в *props* диалогового окна поля для ввода информации об абоненте.

```

addSub() {
    this.$store.commit('subsFrame/setIsAddOperation', true)
    this.modalShow = true

    var eDate = new Date();
    eDate.setMonth(eDate.getMonth() + 1);

    this.modalInfo = {
        id: '',
        subNumber: '',
        fio: '',
        phoneNum: '',
        type: '',
        trainer: '',
        begDate: convert(new Date()),
        endDate: convert(eDate),
        begTime: '',
        trainLeft: '',
        payLeft: '',
        note: ''
    }
}

```

Метод *onRowClicked* запускается по нажатию пользователем на абонент из списка, хранящегося в таблице. Делает диалоговое окно просмотра абонента видимым.

```

onRowClicked(idx) {

```

```

    this.modalInfo = this.subList[idx];
    this.receivedSubNumber = this.subList[idx].subNumber
    this.modalShow = true
  },

```

Метод *search* – асинхронный метод, выполняющий запрос к серверу для поиска абонементов. Для поиска использует пользовательский ввод и критерий для поиска. Делает запрос к серверу при помощи *axios*. Результат запроса – список абонементов помещается в *subList* и отображается в таблице из-за реактивности свойства *computed*.

```

async search() {
  let res

  this.subList = [];

  if(!this.userInput) {
    res = await this.$axios.get('https://localhost:3000/v1/subs/getLatest' , {
      headers: {
        Authorization: "Bearer " + this.$store.getters.getToken
      }
    })
  } else if(this.searchCriterion === 'Номер абонемента') {
    res = await
this.$axios.post('https://localhost:3000/v1/subs/getSubBySubNumber', {
      sub_number: this.userInput
    } , {
      headers: {
        Authorization: "Bearer " + this.$store.getters.getToken
      }
    })
  } else if(this.searchCriterion === 'ФИО') {
    res = await this.$axios.post('https://localhost:3000/v1/subs/getSubByFio', {
      fio: this.userInput
    } , {
      headers: {
        Authorization: "Bearer " + this.$store.getters.getToken
      }
    })
  } else {
    res = await
this.$axios.post('https://localhost:3000/v1/subs/getSubByPhoneNumber', {
      phone_number: this.userInput
    } , {
      headers: {
        Authorization: "Bearer " + this.$store.getters.getToken
      }
    })
  }

  res.data.forEach(element => {
    this.subList.push({
      subId: element.sub_id,

```

```

        clientId: element.client_id,
        trainerId: element.trainer_id,
        typeId: element.type_id,
        subNumber: element.sub_number,
        fio: element.client_fio,
        phoneNum: element.phone_number,
        type: element.title ? (element.title + ' ' + element.training + ' ' +
trainCase(element.training) + ' ' + element.cost + ' рублей') : '',
        trainer: element.trainer_fio,
        begDate: convert(element.begin_date),
        endDate: convert(element.end_date),
        begTime: element.start_time,
        trainLeft: element.training_left,
        payLeft: element.left_to_pay,
        note: element.note,
        client: element.client_fio + ' ' + element.phone_number,
        oldTypeId: element.type_id,
    })
});

this.subList = this.subList.reverse()
},

```

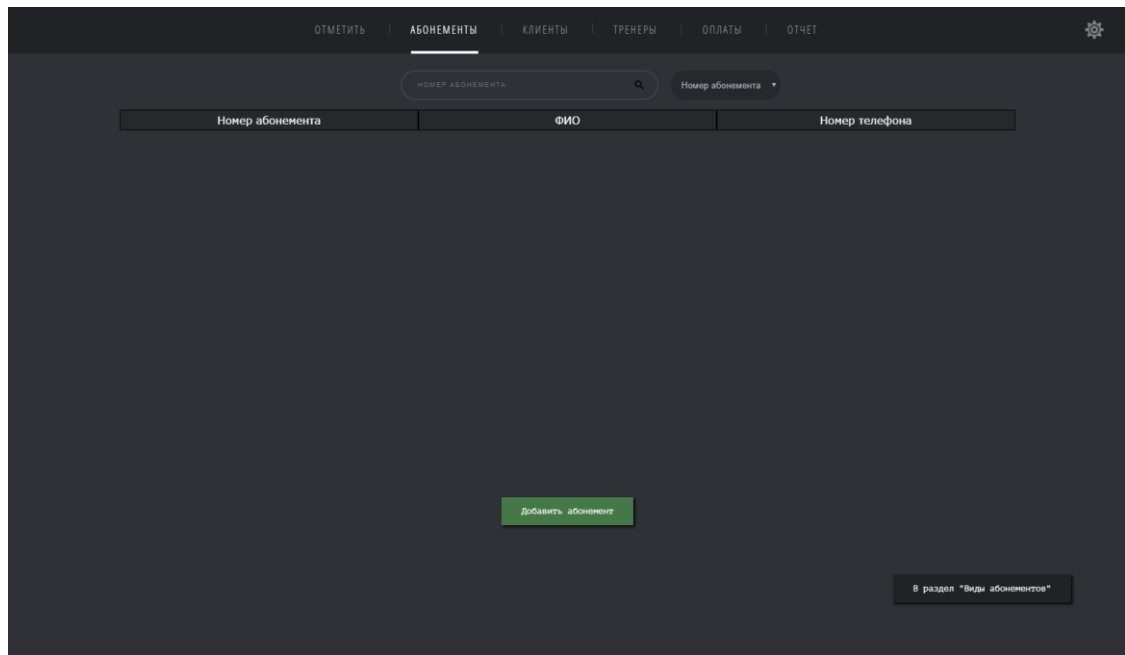


Рисунок 20 – Окно «Абонементы»

Ниже представлено описание окна «Просмотр абонемента» с соответствующими пояснениями. Внешний вид окна представлен на рисунке 21.

Окно состоит из нескольких вложенных блоков (div). Первый блок – *main-shadow*, отвечающий за создание затемнения фона при открытии диалогового окна. Позволяет выделить окно на общем фоне и обеспечивает лучшее

восприятие информации. Внутри *main-shadow* находится блок *main-form*, внутри которого расположены компоненты, хранящие информацию об абонементе и кнопки для манипуляции данными.

```
<template>
  <div class="main-shadow">
    <div class="main-form">
      <div class="main-modal">
        <div class="close-button" @click="close">x</div>
        <div class="info-wrapper">
          <div class="static-info-rows">
            <div v-for="item in gridRowsToShow" :key="item">
              {{ item }}
            </div>
          </div>

          <div class="dynamic-info-rows">
            <input
              class="user-input"
              v-for="(value, key) in gridNodesToShow"
              :key="key"
              type="text"
              :class="{
                'bordered': (isEditOperation || isAddOperation) && key
                !== 'begDate' && key !== 'endDate' && key !== 'payLeft',
                'disabled-border': (isEditOperation || isAddOperation)
                && (key === 'begDate' || key === 'endDate' || key === 'payLeft' || key === 'trainLeft')
              }"
              :disabled="!(isEditOperation || isAddOperation) || key ===
                'begDate' || key === 'endDate' || key === 'payLeft' || key === 'trainLeft'"
              v-model="gridNodes[key]"
              @click="inputClicked(key)"
            />
            <div v-show="isAddOperation" class="date-pick-wrapper">
              <date-picker
                range
                v-model="pickedDate"
                @input="datePicked"
              />
            </div>
          </div>

          <div class="add-button-wrapper" v-if="isAddOperation">
            <button class="confirm-add-type-button"
              @click="addSub">Добавить</button>
          </div>

          <div class="edit-button-wrapper" v-if="!isAddOperation">
            <button class="edit-type-button" @click="editSub">{{
              isEditOperation ? 'Применить' : 'Изменить' }}</button>
            <button class="extend-type-button"
              @click="extendSub">Продлить</button>
            <button class="remove-type-button"
              @click="removeSubClicked">Удалить</button>
          </div>
        </div>
      </div>
    </div>
  </div>
```

```

    </div>

    <confirmModal
      v-show="confirmVisible"
      @agreeClose="removeSub"
      @disagreeClose="confirmVisible = false"
      v-bind:questionString="'Удалить абонемент?'"
    />

    <helperModal
      v-show="helperVisible"
      v-bind:helperTitle="helperTitle"
      v-bind:currentOptionKey="currentOptionKey"
      :key="searchPanelKey"
      @modalClose="helperVisible = false"
      @rowChooosed="rowChooosed"
    />

    <extend-modal
      v-show="extendVisible"
      v-bind:begDate="gridNodes.begDate"
      v-bind:endDate="gridNodes.endDate"
      @extendClose="extendClose"
      @extendConfirmed="extendConfirmed"
    />
  </div>
</div>
</div>
</template>

```

В скрипте импортируются нужные компоненты (выбора дат, окно подтверждения, окно для выбора клиента/тренера/вида абонемента, хранилище) и скрипты (валидация, приведение слов в нужный падеж)

```

import DatePicker from 'vue2-datepicker';
import 'vue2-datepicker/index.css';

import confirmModal from './confirmModal'
import helperModal from './addSubHelperModal'
import extendModal from './extendSubModal'

import { mapGetters } from 'vuex'
import validate from '../../validation/subValidation'
import trainCase from '../../util/trainingCase'

```

Для получения данных из родительский окон Vue.js реализует свойство `props` экземпляра Vue. Из родительского окна получаем данные для работы с абонементом.

```

props: {
  gridRows: Array,
  gridNodes: Object,

```

```

        receivedSubNumber: String
    },

```

Для редактирования абонемента предусмотрены переменные *choosedId*, хранящий выбранные пользователем данные клиента, тренера и вида абонемента и *pickedDate*, хранящий выбранные пользователем даты действия абонемента.

```

data() {
    return {
        confirmVisible: false,
        helperVisible: false,
        helperTitle: '',
        currentOptionKey: 'clients',
        searchPanelKey: 0,
        choosedId: {
            clientId: '',
            trainerId: '',
            typeId: ''
        },
        extendVisible: false,
        pickedDate: ''
    }
},

```

Компонента *date-picker* вызывает метод *datePicked* при выборе пользователем диапазона дат. Внешний вид окна представлен на рисунке 23.

```

datePicked(date) {
    var mnth = ("0" + (this.pickedDate[0].getMonth() + 1)).slice(-2)
    var day = ("0" + this.pickedDate[0].getDate()).slice(-2)
    this.gridNodes.begDate = [day, mnth,
this.pickedDate[0].getFullYear()].join(".")

    var mnth = ("0" + (this.pickedDate[1].getMonth() + 1)).slice(-2)
    var day = ("0" + this.pickedDate[1].getDate()).slice(-2)
    this.gridNodes.endDate = [day, mnth,
this.pickedDate[1].getFullYear()].join(".")
}

```

Для изменения абонемента используется метод *editSub*. Отправляет запрос на изменения абонемента на сервер, используя данные с переменных пользовательского ввода. Внешний вид окна представлен на рисунке 22.

```

async editSub() {
    if (this.isEditOperation) {

```

```

        var { isCorrect, alertMessage } = await validate(this.gridNodes,
this.isEditOperation)

        // Если при изменении абонента номер абонента не был изменен,
        // проверять уникальность не требуется
        if(this.gridNodes.subNumber != this.receivedSubNumber) {
            let res = await
this.$axios.post('https://localhost:3000/v1/subs/checkUniq', {
                sub_number: this.gridNodes.subNumber
            }, {
                headers: {
                    Authorization: "Bearer " + this.$store.getters.getToken
                }
            })

            if(!res.data) {
                alertMessage = '• Такой номер абонента уже существует\n' +
alertMessage
                isCorrect = false
            }
        }

        if(!isCorrect) {
            alert(alertMessage)
            return
        }

        var begSplit = this.gridNodes.begDate.split(/[-/]/);
        var endSplit = this.gridNodes.endDate.split(/[-/]/);

        var begDateToSend = new Date(begSplit[2], begSplit[1] - 1,
parseInt(begSplit[0]) + 1);
        var endDateToSend = new Date(endSplit[2], endSplit[1] - 1,
parseInt(endSplit[0]) + 1);

        var clientIdToSend = this.choosedId.clientId ? this.choosedId.clientId
: this.gridNodes.clientId
        var trainerIdToSend = this.choosedId.trainerId ?
this.choosedId.trainerId : this.gridNodes.trainerId
        var typeIdToSend = this.choosedId.typeId ? this.choosedId.typeId :
this.gridNodes.typeId

        await this.$axios.post('https://localhost:3000/v1/subs/edit', {
            id: this.gridNodes.subId,
            sub_number: this.gridNodes.subNumber,
            type_id: typeIdToSend,
            client_id: clientIdToSend,
            trainer_id: trainerIdToSend,
            begin_date: begDateToSend,
            end_date: endDateToSend,
            start_time: this.gridNodes.begTime + ':00',
            training_left: this.gridNodes.trainLeft,
            left_to_pay: this.gridNodes.payLeft,
            note: this.gridNodes.note,
            old_type: this.gridNodes.oldTypeId
        }, {
            headers: {
                Authorization: "Bearer " + this.$store.getters.getToken
            }
        })
    }
}

```

```

    }
  })

  this.$emit('modalClose');
}
this.$store.commit('subsFrame/setIsEditOperation', !this.isEditOperation)
},

```

Для добавления абонента используется метод *addSub*. Отправляет запрос на изменения абонента на сервер, используя данные с переменных пользовательского ввода. Внешний вид окна представлен на рисунке 25.

```

async addSub() {
  var { isCorrect, alertMessage } = await validate(this.gridNodes,
this.isEditOperation)

  let res = await
this.$axios.post('https://localhost:3000/v1/subs/checkUniq', {
    sub_number: this.gridNodes.subNumber
  }, {
    headers: {
      Authorization: "Bearer " + this.$store.getters.getToken
    }
  })

  if(!res.data) {
    alertMessage = '• Такой номер абонента уже существует\n' +
alertMessage
    isCorrect = false
  }

  if(!isCorrect) {
    alert(alertMessage)
    return
  }

  var begSplit = this.gridNodes.begDate.split(/[-]/);
  var endSplit = this.gridNodes.endDate.split(/[-]/);

  var begDateToSend = new Date(begSplit[2], begSplit[1] - 1,
parseInt(begSplit[0]) + 1);
  var endDateToSend = new Date(endSplit[2], endSplit[1] - 1,
parseInt(endSplit[0]) + 1);

  await this.$axios.post('https://localhost:3000/v1/subs/add', {
    sub_number: this.gridNodes.subNumber,
    type_id: this.choosedId.typeId,
    client_id: this.choosedId.clientId,
    trainer_id: this.choosedId.trainerId,
    begin_date: begDateToSend,
    end_date: endDateToSend,
    start_time: this.gridNodes.begTime + ':00',
    note: this.gridNodes.note
  }, {
    headers: {
      Authorization: "Bearer " + this.$store.getters.getToken
    }
  })
}

```



```

    })

    this.$store.commit('subsFrame/setIsAddOperation', false)
    this.$emit('modalClose');
  },

```

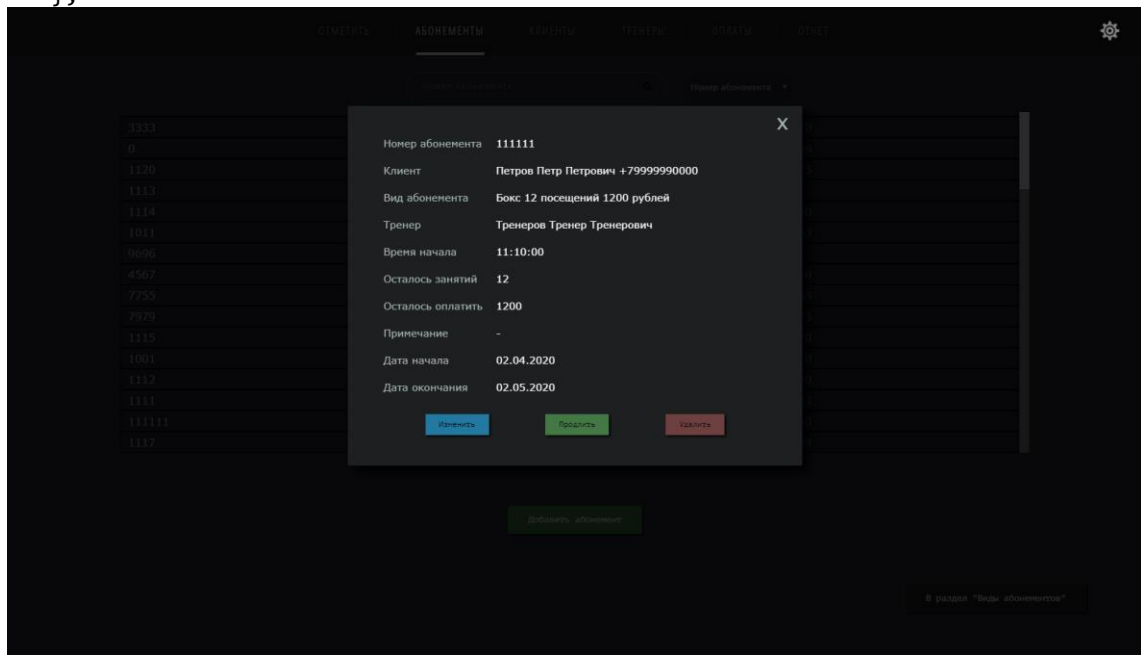


Рисунок 21 – Просмотр информации об абонементе

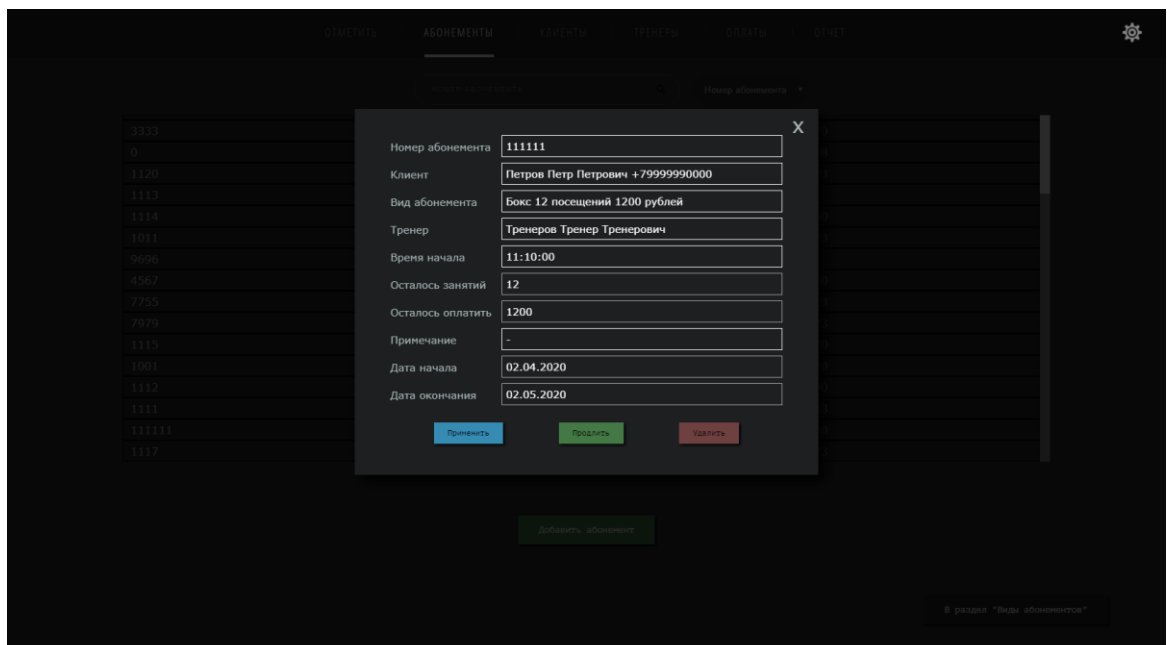


Рисунок 22 – Редактирование абонеента

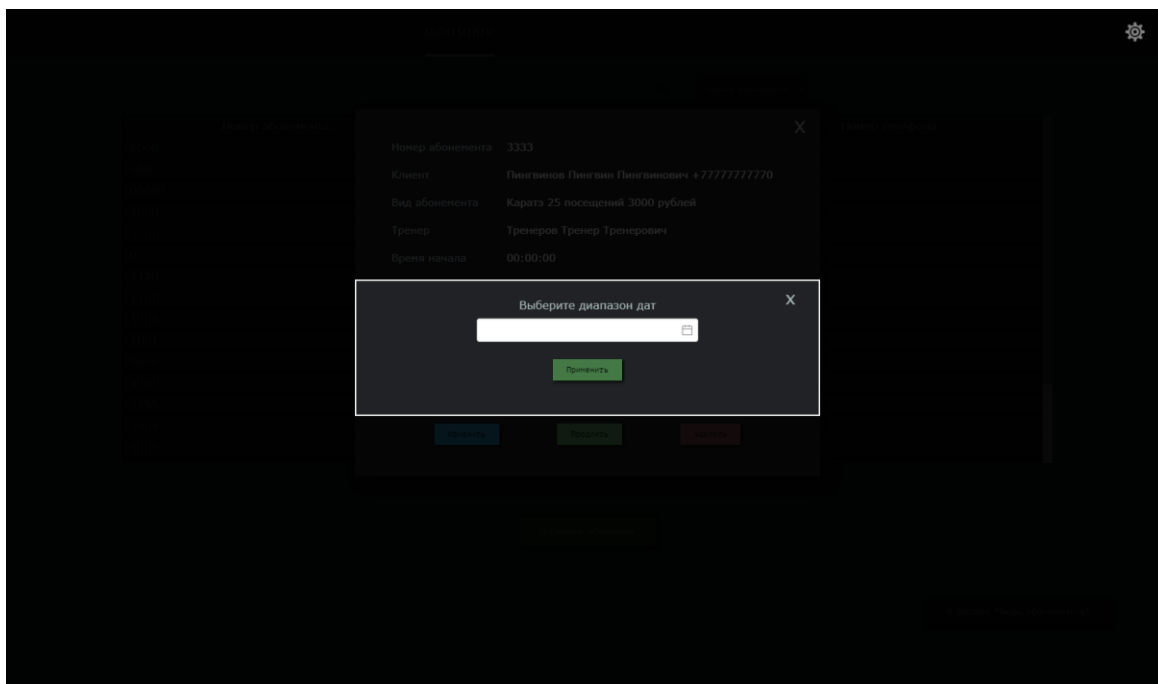


Рисунок 23 – Продление абонемента

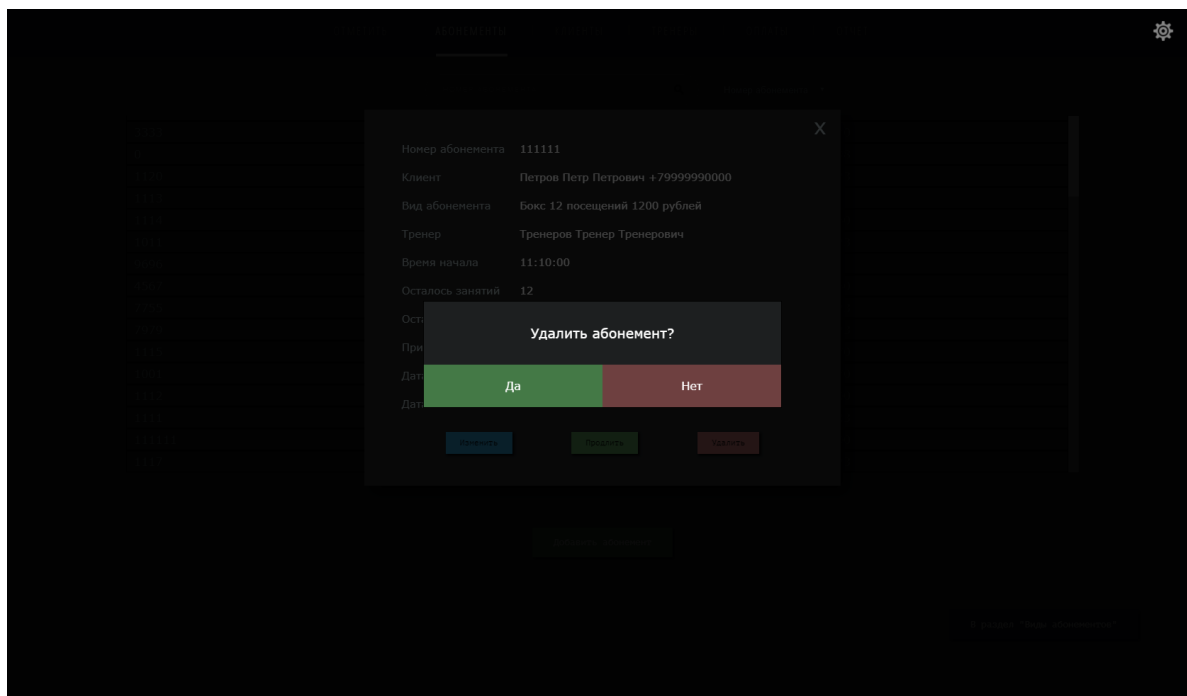


Рисунок 24 – Удаление абонемента

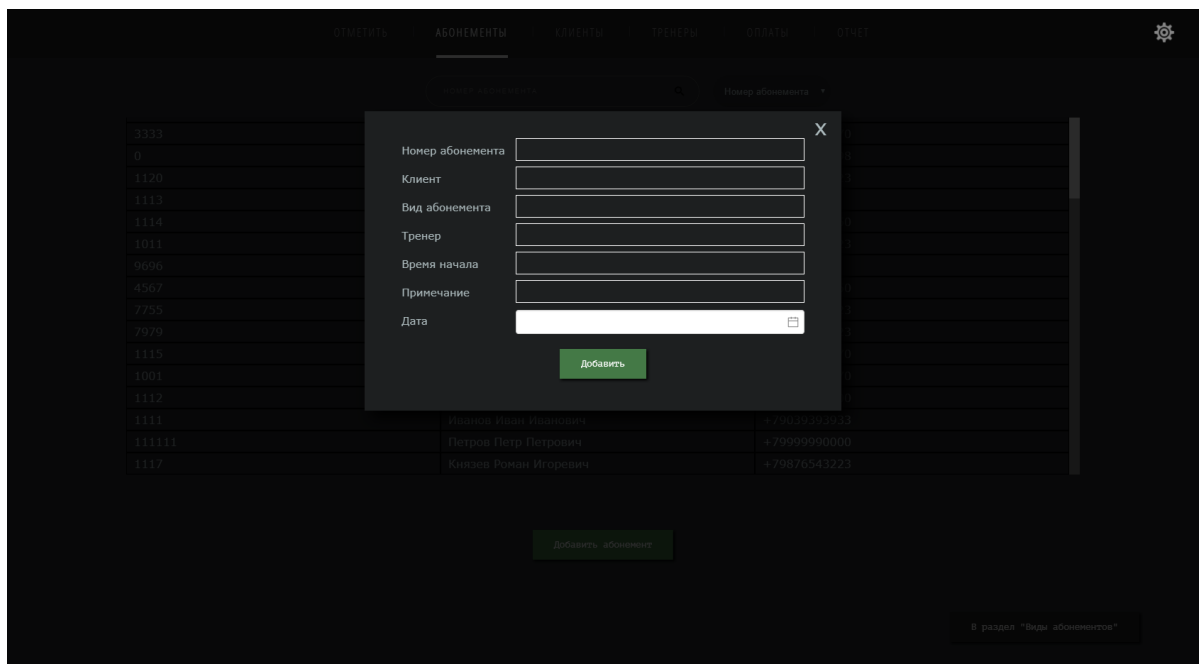


Рисунок 25 – Добавление абонента

При добавлении/изменении абонента применяется валидация для проверки корректности введенных пользователем данных. Ниже представлено описание валидации.

```
import dateValidation from './dateValidation'

export default async function(gridNode, isEditOperation) {
  var isCorrect = true
  var alertMessage = ''
  // Введен ли номер абонента
  if(!gridNode.subNumber) {
    alertMessage += '• Введите номер абонента\n'
    isCorrect = false
  }
  // Проверка длины номера абонента
  } else if(gridNode.subNumber.length > 13) {
    alertMessage += '• Слишком длинный номер абонента\n'
    isCorrect = false
  }

  // Указан ли клиент
  if(!gridNode.client) {
    alertMessage += '• Выберите клиента\n'
    isCorrect = false
  }

  // Указан ли вид абонента
  if(!gridNode.type) {
```

```

        alertMessage += '• Выберите вид абонемента\n'
        isCorrect = false
    }

    // Указан ли тренер
    if(!gridNode.trainer) {
        alertMessage += '• Выберите тренера\n'
        isCorrect = false
    }

    // Указана ли дата начала
    if(!gridNode.begDate) {
        alertMessage += '• Введите дату начала\n'
        isCorrect = false
    }
    // Соответствует ли дата начала формату даты (проверка регулярным выражением)
    } else if(gridNode.begDate && !dateValidation(gridNode.begDate)) { //
        alertMessage += '• Дата начала не соответствует формату (дд.мм.гггг)\n'
        isCorrect = false
    }
    // Дата начала не может быть больше даты окончания
    } else if(gridNode.endDate && gridNode.begDate > gridNode.endDate) {
        alertMessage += '• Дата начала не может быть больше даты окончания\n'
        isCorrect = false
    }
    // Дата окончания
    if(!gridNode.endDate) {
        alertMessage += '• Введите дату окончания\n'
        isCorrect = false
    }
    } else if(!dateValidation(gridNode.endDate)) {
        alertMessage += '• Дата окончания не соответствует формату (дд.мм.гггг)\n'
        isCorrect = false
    }
    // Проверка времени начала на соответствие регулярному выражению в формате чч:мм
    var timeReg = /([0-1][0-9]|2[0-3])([:])([0-5][0-9])/gi
    var regRes = String(gridNode.begTime).match(timeReg)
    if(gridNode.begTime && !regRes) {
        alertMessage += '• Время не соответствует формату (чч:мм)\n'
        isCorrect = false
    } else {
        gridNode.begTime = regRes
    }
    // Не превышает ли длина примечание 200 символов
    if(gridNode.note.length > 200) {
        alertMessage += '• Слишком длинное примечание\n'
        isCorrect = false
    }
}

if(isEditOperation) {
    var trainLeftInt = parseInt(gridNode.trainLeft)
    var payLeftInt = parseInt(gridNode.payLeft)

    // Заполнено ли поле "осталось оплатить"
    if(!gridNode.payLeft === '') {
        alertMessage += "• Заполните поле 'Осталось оплатить'\n"
        isCorrect = false
    }
    // Проверка за выход из целочисленного диапазона для типа integer в БД
    } else if(isNaN(payLeftInt) || payLeftInt < 0 || payLeftInt > 2000000000) {
        alertMessage += "• Некорректное значение поля 'Осталось оплатить'\n"
        isCorrect = false
    }
}

```

```

    }

    // Заполнено ли поле "осталось занятий"
    if(gridNode.trainLeft === '') {
        alertMessage += "• Заполните поле 'Осталось занятий'\n"
        isCorrect = false
    }
    // Проверка за выход из пределов разумного
    } else if(isNaN(trainLeftInt) || trainLeftInt < 0 || trainLeftInt > 30000) {
        alertMessage += "• Некорректное значение поля 'Осталось занятий'\n"
        isCorrect = false
    }
}

if(isCorrect) {
    fillSpaces(gridNode)
}
return { isCorrect, alertMessage }
}

// Автозаполнение пустых необязательных полей
const fillSpaces = function(gridNode) {
    if(!gridNode.note) {
        gridNode.note = '-'
    }
    if(!gridNode.begTime) {
        gridNode.begTime = '00:00'
    }
}
}

```

## Реализация отчета «Начисления при продаже»

Окно содержит панель фильтров и таблицу оплат. Имеется возможность фильтрации по тренеру, датам и времени начала. Время позволяет отделять одно групповое занятие от другого.

Окно отчета «Начисления при продаже» представлено на рисунке 26.

ОТМЕТИТЬ   АБОНЕМЕНТЫ   КЛИЕНТЫ   ТРЕНЕРЫ   ОПЛАТЫ   ОТЧЕТ							
Тренер Все		Диапазон дат		Время начала НН:мм			
Тренер	Номер абонемента	Вид абонемента	Дата оплаты	Сумма оплаты	Ставка тренера	Начисление тренеру	Время начала
Бицисов Трицепс Квадрицепсович	1010	Каратэ 25 посещений 3000 рублей	30.05.2020	9000	50	4500	00:00
Бицисов Трицепс Квадрицепсович	4567	Мультиабонемент 2 30 посещений 10000 рублей	30.05.2020	500	50	250	00:00
Дунаев Никита Юрьевич	00000	Кикбоксинг 8 посещений 3000 рублей	12.02.2020	3000	40	1200	00:00
Дунаев Никита Юрьевич	0	Каратэ 25 посещений 3000 рублей	02.05.2020	2998	40	1199.2	00:00
Дунаев Никита Юрьевич	00000	Кикбоксинг 8 посещений 3000 рублей	30.04.2020	500	40	200	00:00
Дунаев Никита Юрьевич	00000	Кикбоксинг 8 посещений 3000 рублей	02.05.2020	1000	40	400	00:00
Дунаев Никита Юрьевич	7979	Сауна 10 посещений 1000 рублей	30.05.2020	1000	50	500	00:00
Бицисов Трицепс Квадрицепсович	6060		30.05.2020	123	50	61.5	00:00
Дунаев Никита Юрьевич	1111	Бокс 12 посещений 1200 рублей	30.05.2020	1500	50	750	13:00
Дунаев Никита Юрьевич	1112	Бокс 12 посещений 1200 рублей	09.05.2020	1000	60	600	13:00

Экспорт в Excel

Сумма по оплатам: 40621  
Сумма по начислениям тренеру: 21660.7

Рисунок 26 – Окно отчета «Начисления при продаже»

Для экспорта отчета в Excel в окне «Начисления при продаже» был реализован метод *toCsv*. В данном методе собирается строка в формате csv, содержащая имена столбцов и преобразующая массив строк отчета в табличный вид путем разделения символами «;» и «\n».

```
toCsv() {
    var csvString = 'Тренер;Номер абонемента;Вид абонемента;Дата оплаты;Сумма
оплаты;Ставка тренера;Начисление тренеру;Время начала\n'

    this.resultList.forEach(element => {
        let line =
            element.fio + ';' +
            element.subNumber + ';' +
            element.subType + ';' +
            element.paymentDate + ';' +
            element.paymentAmount + ';' +
            element.interestRate + ';' +
            element.trainerSalary + ';' +
            element.begTime + '\n'

        csvString += line
    });

    var textEncoder = new TextEncoder('windows-1252');
    var csvContentEncoded = textEncoder.encode([csvString]);
    var blob = new Blob([csvContentEncoded], {type: 'text/csv;charset=windows-
1252;'}));
    var url = window.URL.createObjectURL(blob);
    var a = document.createElement("a");
    document.body.appendChild(a);
}
```

```

a.style = "display: none";
a.href = url;
a.download = 'report.csv'
a.click();

document.body.removeChild(a);
},

```

Результат экспорта в Excel представлен на рисунке 27.

Тренер	Номер абонемента	Вид абонемента	Дата оплаты	Сумма оплаты	Ставка тренера	Начисление тренеру	Время начала
Бицелсов Трицепс Квадрицепсович	1010	Карата 25 посещений 3000 рублей	30.05.2020	9000	50	4500	0:00:00
Дунаев Никита Юрьевич	4567	Мультиабонемент 2 30 посещений 10000 рублей	30.05.2020	500	50	250	0:00:00
Дунаев Никита Юрьевич	0	Кинбоксинг 8 посещений 3000 рублей	12.02.2020	3000	40	1200	0:00:00
Дунаев Никита Юрьевич	0	Карата 25 посещений 3000 рублей	02.05.2020	2998	40	1199.2	0:00:00
Дунаев Никита Юрьевич	0	Кинбоксинг 8 посещений 3000 рублей	30.04.2020	500	40	200	0:00:00
Дунаев Никита Юрьевич	0	Кинбоксинг 8 посещений 3000 рублей	02.05.2020	1000	40	400	0:00:00
Дунаев Никита Юрьевич	7979	Сауна 10 посещений 1000 рублей	30.05.2020	1000	50	500	0:00:00
Бицелсов Трицепс Квадрицепсович	6060		30.05.2020	123	50	61.5	0:00:00
Дунаев Никита Юрьевич	1111	Бокс 12 посещений 1200 рублей	30.05.2020	1500	50	750	13:00:00
Дунаев Никита Юрьевич	1112	Бокс 12 посещений 1200 рублей	09.05.2020	1000	60	600	13:00:00
Бицелсов Трицепс Квадрицепсович	666	Годовая качалка 1200 посещений 20000 рублей	02.05.2020	20000	60	12000	14:00:00

Рисунок 27 – Отчет в Excel

## 8 Тестирование

Тестирование системы учета производилось в функциональном режиме. Для наиболее полного и качественного тестирования, были разработаны тест-кейсы, которые позволяют продумать проверку, как бизнес логики, так и технической части. Тест-кейс - описание проверки с определенным набором шагов. Хороший кейс должен быть таким, чтобы протестировать описанных функционал мог любой член команды, будь то разработчик, аналитик или продукт менеджер. Он должен быть понятным и тщательно описан. Тест-кейс состоит из:

- уникального идентификатора;
- названия - краткое описание сути проверки;
- описания тестируемого функционала - полное описание сути проверки;
- предусловия - действие, которое должно быть выполнено до начала тестирования, но прямого отношения к проверке они не имеют;
- шагов - описание действий проверки;
- ожидаемый результат - результат, который должен являться корректным.

### Тест 1. Добавление клиента

#### *Предусловия:*

- пользователь должен быть авторизован.

#### *Шаги:*

- перейти в окно «Клиенты»;
- нажать кнопку «Добавить клиента»;
- заполнить поля ввода и сделать фотографию (действие представлено на рисунке 28);
- нажать кнопку «Добавить».

#### *Ожидаемый результат:*



- в окне «Клиенты» появилась запись о новом клиенте, что показано на рисунке 29.

Рисунок 28 – Заполнение полей для добавления клиента

ФИО	Номер телефона
Дунаев Никита Юрьевич	+79969969696

Рисунок 29 – Результат добавления клиента.

*Результат: пройден.*

## Тест 2. Редактирование данных клиента.

### Предусловия:

- пользователь должен быть авторизован;
- должна быть запись о клиенте.

### Шаги:

- перейти в окно «Клиенты»;
- нажать на клиента в таблице;
- нажать кнопку «Изменить»;
- изменить поля (действие представлено на рисунке 30);
- нажать кнопку «Применить».

### Ожидаемый результат:

- информация о клиенте будет обновлена.

The screenshot displays a web interface with a top navigation bar containing tabs: ОТМЕНА, АРХИВ, КЛИЕНТЫ, ТИПЫ, ОБРАТЫ, and ОТЧЕТ. The 'КЛИЕНТЫ' tab is active. Below the navigation bar, there is a table with two columns: 'ФИО' and 'Номер телефона'. The first row shows 'Дунаев Никита Юрьевич' and '+9500000000'. A modal window for editing the client is open in the center. It has a close button (X) in the top right corner. The modal contains the following fields: 'ФИО' (Dunaev Nikita Yuryevich), 'Номер телефона' (+9500000000), 'Откуда узнали' (Яндекс), 'Кто пригласил' (empty), 'Примечание' (AaAa), and 'Дата первого визита' (2020-05-07). There is a video preview window showing a person holding a red envelope. Below the fields are buttons for 'Применить' (Apply), 'Удалить' (Delete), and 'Отменить' (Cancel). At the bottom of the modal is a button 'Добавить клиента' (Add client).

Рисунок 30 - Редактирование данных клиента

**Результат: пройден.**

### Тест 3. Удаление клиента

#### Предусловия:

- пользователь должен быть авторизован;
- должна быть запись о клиенте.

#### Шаги:

- перейти в окно «Клиенты»;
- нажать на клиента в таблице;
- нажать кнопку «Удалить»;
- подтвердить удаление.

#### Ожидаемый результат:

- запись о клиенте удалена, что показано на рисунке 31.

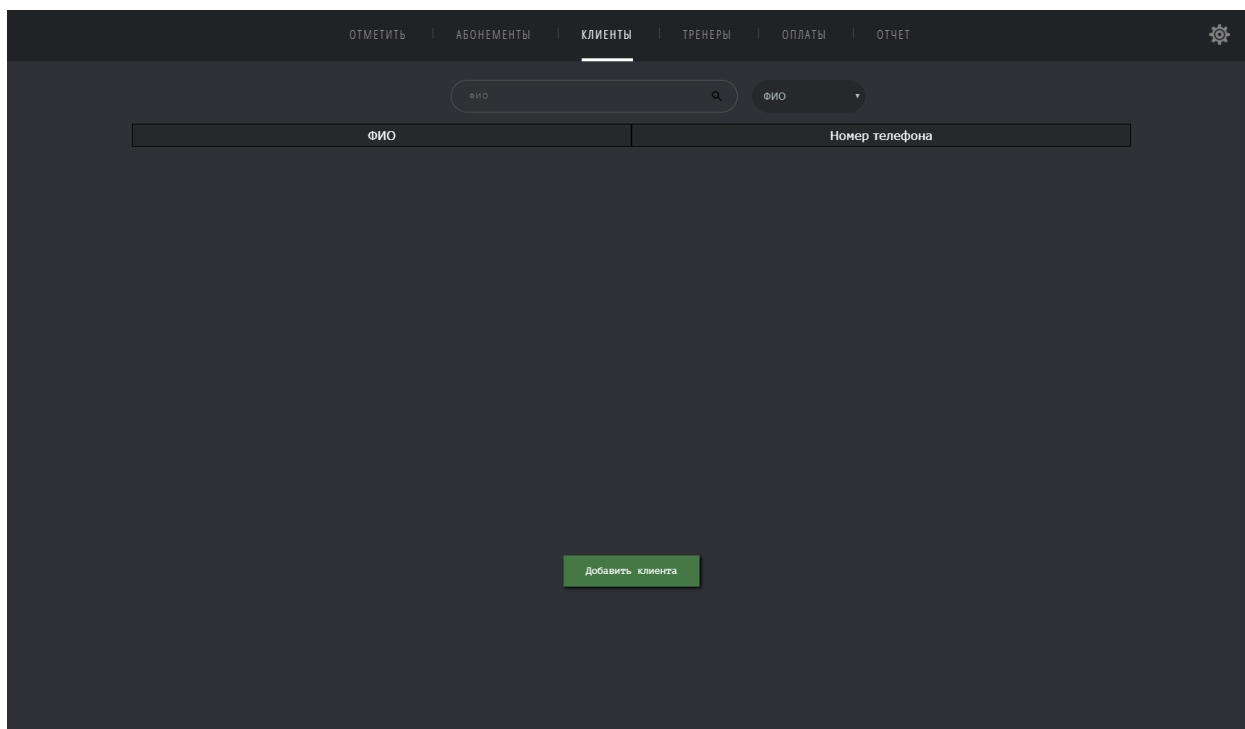


Рисунок 31 – Результат удаления клиента

*Результат:* **пройден.**

## Тест 4. Поиск клиента

### Предусловия:

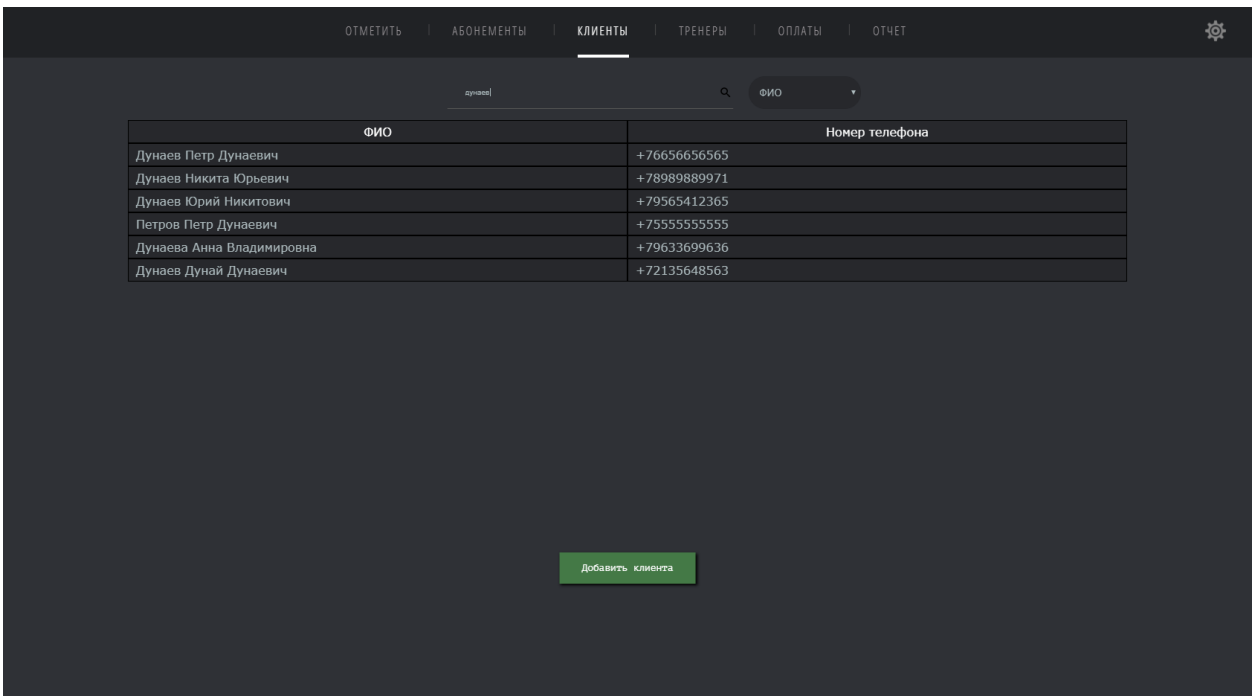
- пользователь должен быть авторизован;
- должна быть запись о клиенте.

### Шаги:

- перейти в окно «Клиенты»;
- выбрать критерий поиска;
- ввести текст поиска.

### Ожидаемый результат:

- получен список клиентов, удовлетворяющий запросу, что показано на рисунке 32.



ФИО	Номер телефона
Дунаев Петр Дунаевич	+76656656565
Дунаев Никита Юрьевич	+78989889971
Дунаев Юрий Никитович	+79565412365
Петров Петр Дунаевич	+75555555555
Дунаева Анна Владимировна	+79633699636
Дунаев Дунай Дунаевич	+72135648563

Добавить клиента

Рисунок 32 - Результат поиска клиента

*Результат: пройден.*

## Тест 5. Добавление Тренера

*Предусловия:*

- пользователь должен быть авторизован.

*Шаги:*

- перейти в окно «Тренеры»;
- нажать кнопку «Добавить тренера»;
- заполнить поля ввода;
- нажать кнопку «Добавить».

*Ожидаемый результат:*

- в окне «Тренеры» появилась запись о новом тренере, что показано на рисунке 33.

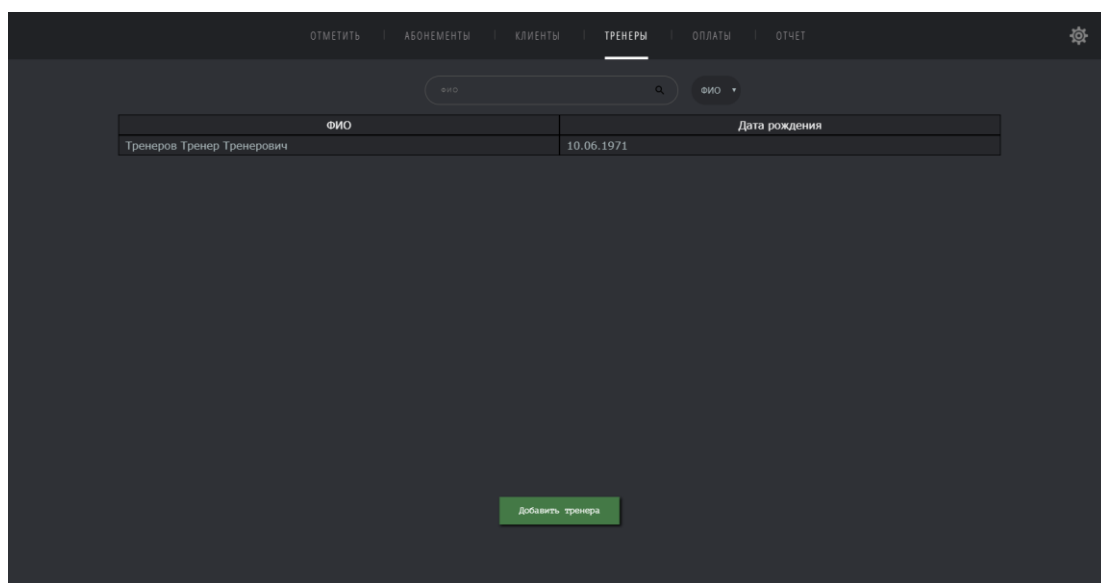


Рисунок 33 - Результат добавления тренера

*Результат: пройден.*

## Тест 6. Редактирование информации тренера

*Предусловия:*

- пользователь должен быть авторизован;

- должна быть запись о тренере.

*Шаги:*

- перейти в окно «Тренеры»;
- нажать на запись о тренере;
- нажать кнопку «Изменить»;
- редактировать поля ввода (действие представлено на рисунке 34);
- нажать кнопку «Применить».

*Ожидаемый результат:*

- информация о тренере поменялась.

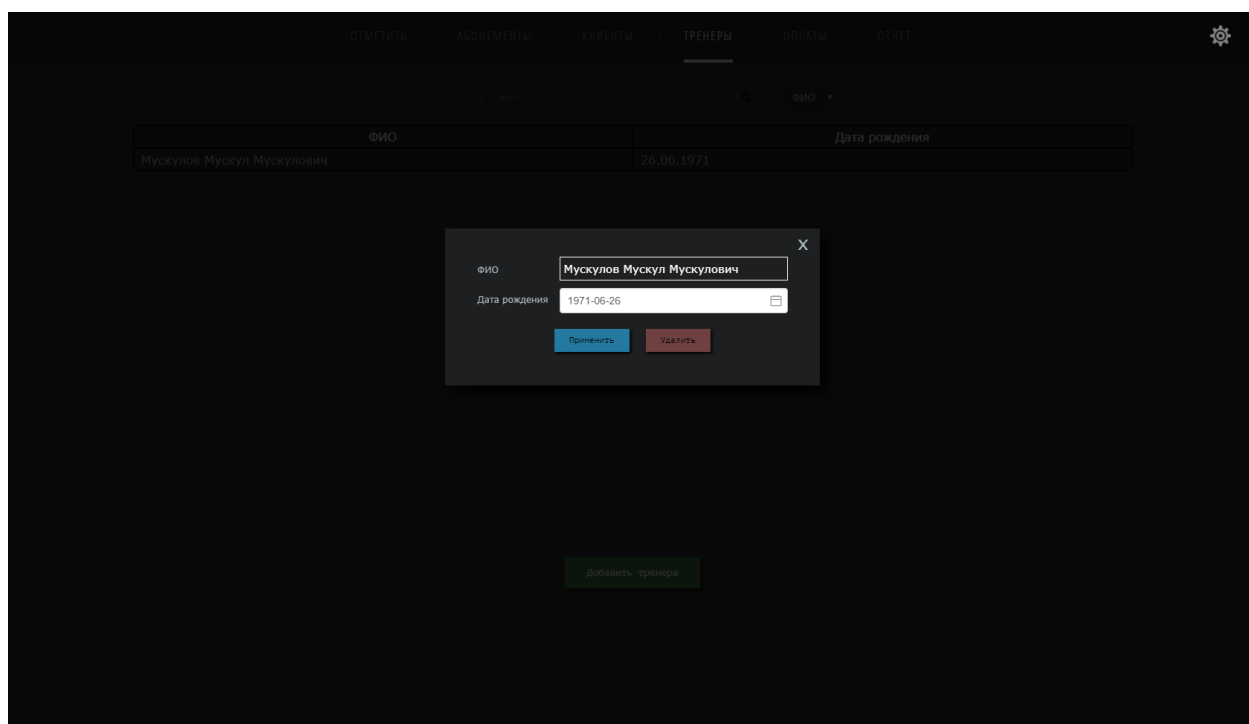


Рисунок 34 – Редактирование информации тренера

*Результат: пройден.*

## Тест 7. Удаление тренера

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись о тренере.

*Шаги:*

- перейти в окно «Тренеры»;
- нажать на запись о тренере;
- нажать кнопку «Удалить»;
- подтвердить выбор.

*Ожидаемый результат:*

- запись о тренере удалена, что показано на рисунке 35.

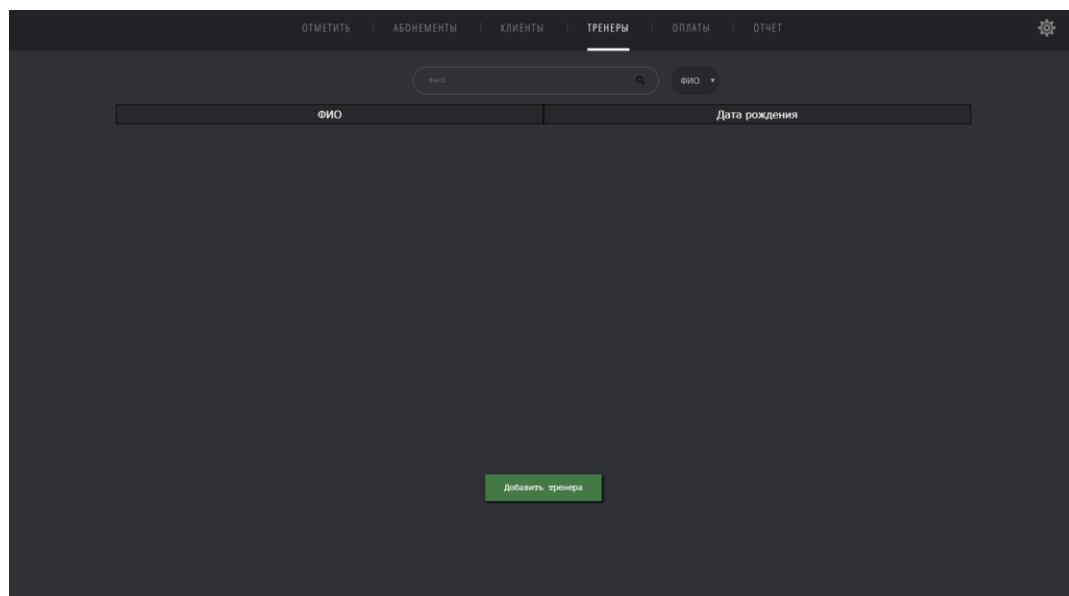


Рисунок 35 – Результат удаления тренера.

*Результат: пройден.*

## **Тест 8. Добавление вида абонеента**

*Предусловия:*

- пользователь должен быть авторизован.

*Шаги:*

- перейти в окно «Абонементы»;
- нажать на кнопку «В раздел «Виды абонементов»»;
- нажать кнопку «Добавить вид абонемента»;
- заполнить поля;
- нажать на кнопку «Добавить».

*Ожидаемый результат:*

- запись о виде абонемента добавлена, что показано на рисунке 36.

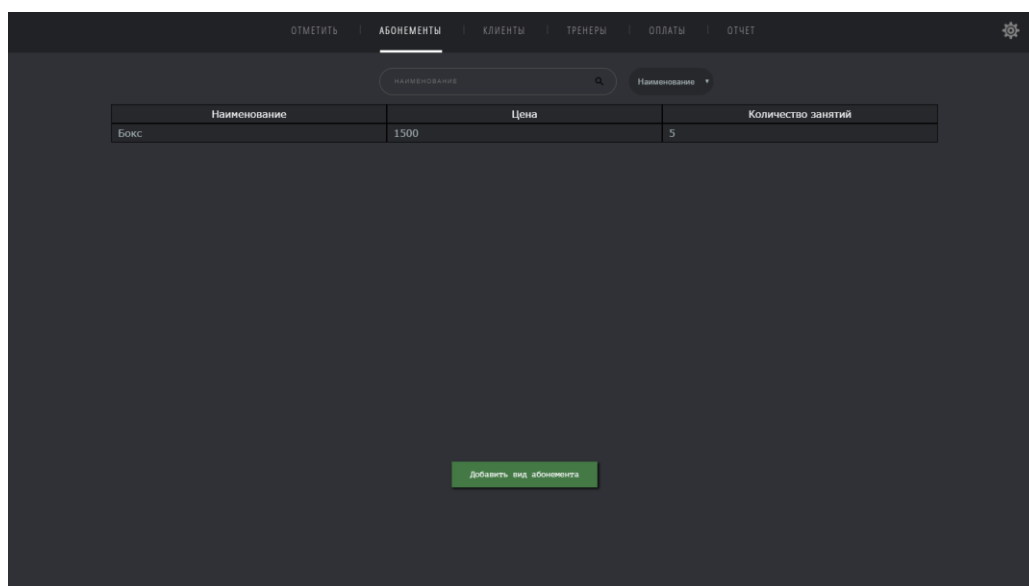


Рисунок 36 – Результат добавления вида абонемента.

*Результат: пройден.*

## Тест 9. Редактирование вида абонемента

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись о виде абонемента.

*Шаги:*

- перейти в окно «Абонементы»;



- нажать на кнопку «В раздел «Виды абонементов»»;
- нажать на запись о виде абонемента;
- нажать на кнопку «Изменить»;
- редактировать поля (действие представлено на рисунке 37);
- нажать на кнопку «Применить»;

*Ожидаемый результат:*

- запись о виде абонемента обновлена.

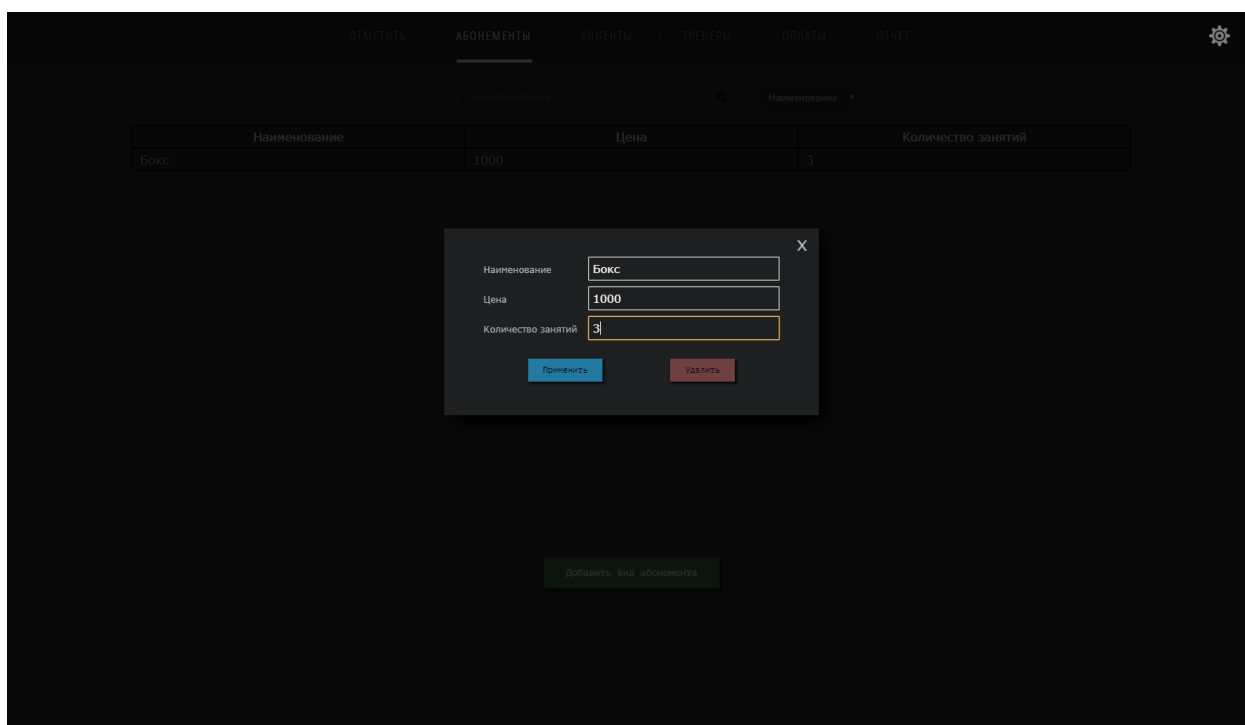


Рисунок 37 – Редактирование вида абонемента

*Результат: пройден.*

#### **Тест 10. Удаление вида абонемента**

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись о виде абонемента.

*Шаги:*

- перейти в окно «Абонементы»;
- нажать на кнопку «В раздел «Виды абонементов»»;
- нажать на запись о виде абонемента;
- нажать на кнопку «Удалить»;
- подтвердить выбор;

*Ожидаемый результат:*

- запись о виде абонемента удалена, что представлено на рисунке 38.

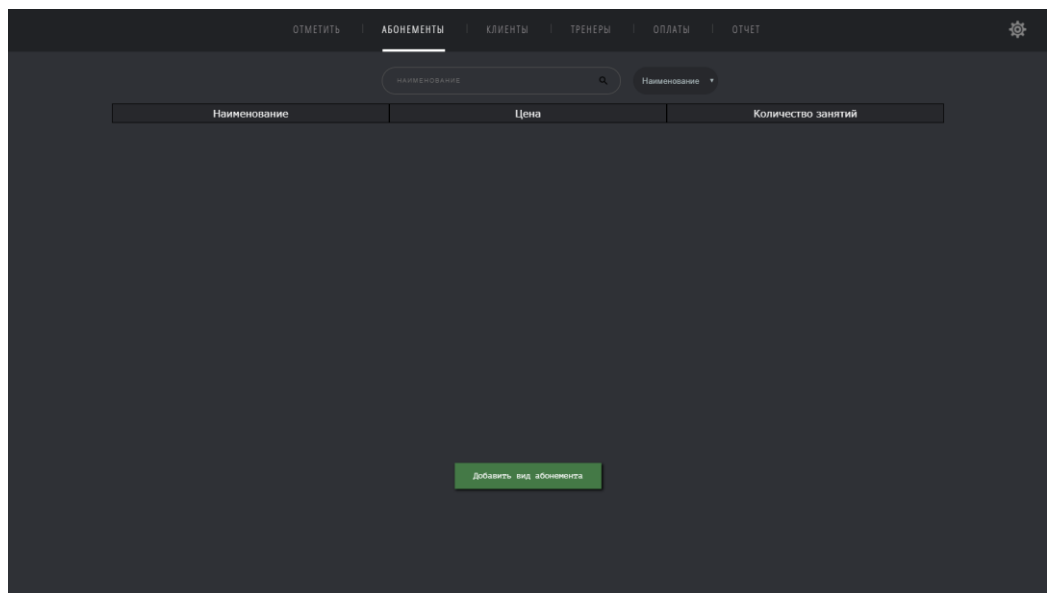


Рисунок 38 – Результат удаления вида абонемента

*Результат: пройден.*

## Тест 11. Добавление абонемента

*Предусловия:*

- пользователь должен быть авторизован.

*Шаги:*

- перейти в окно «Абонементы»;

- нажать кнопку «Добавить абонемент»;
- ввести уникальный номер абонемента;
- выбрать клиента из списка (действие представлено на рисунке 39);
- выбрать вид абонемента из списка (действие представлено на рисунке 40);
- выбрать тренера из списка (действие представлено на рисунке 41);
- заполнить остальные поля (действие представлено на рисунке 42);
- нажать кнопку «Добавить».

*Ожидаемый результат:*

- запись об абонементе добавлена.

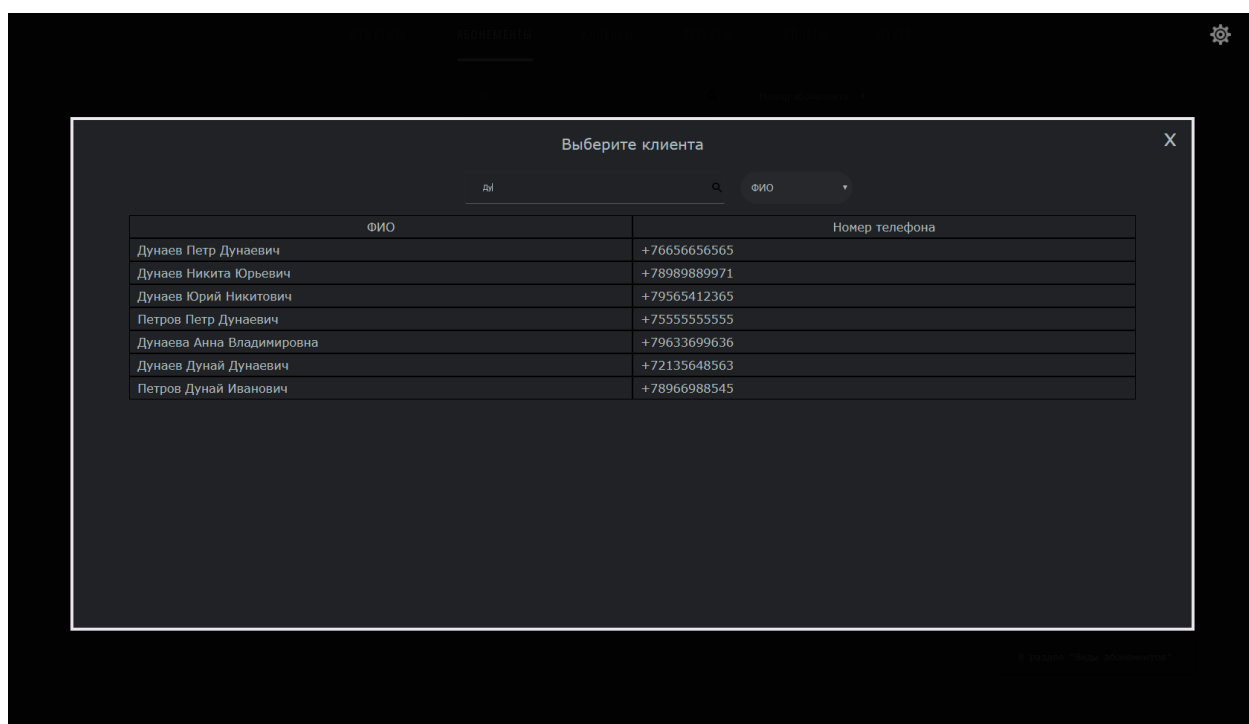


Рисунок 39 – Выбор клиента для добавления абонемента

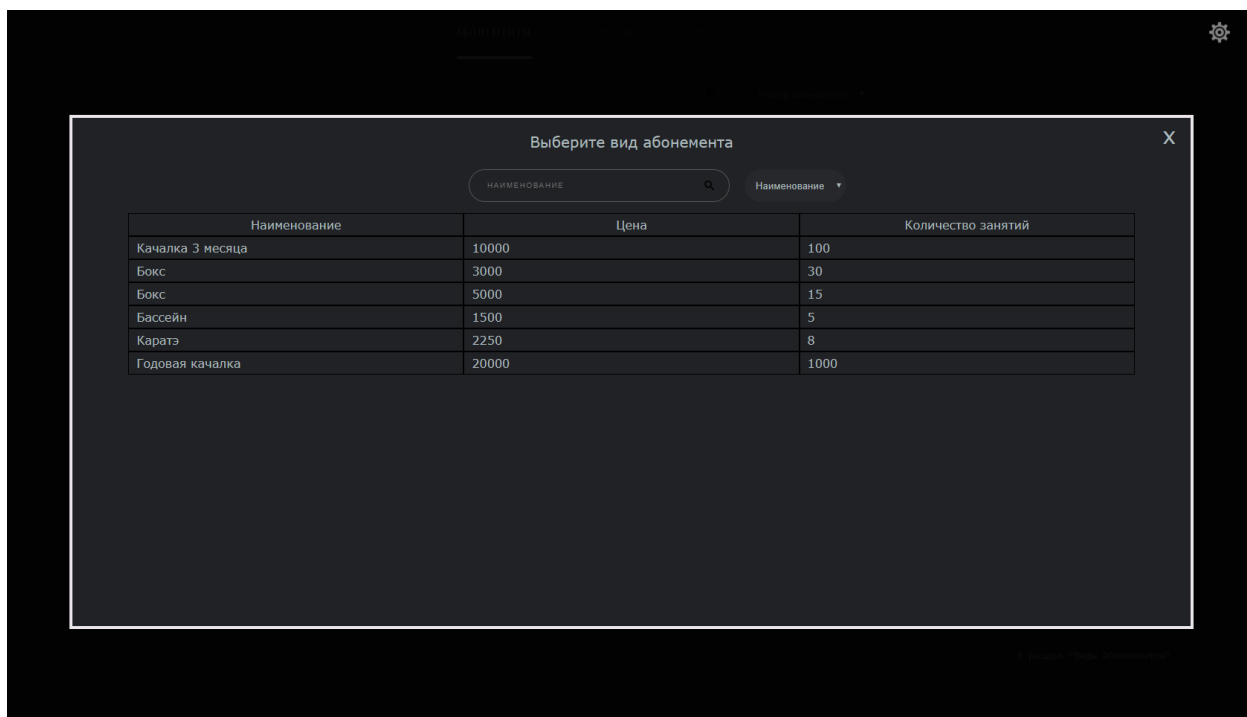


Рисунок 40 – Выбор вида абонеента для добавления абонеента

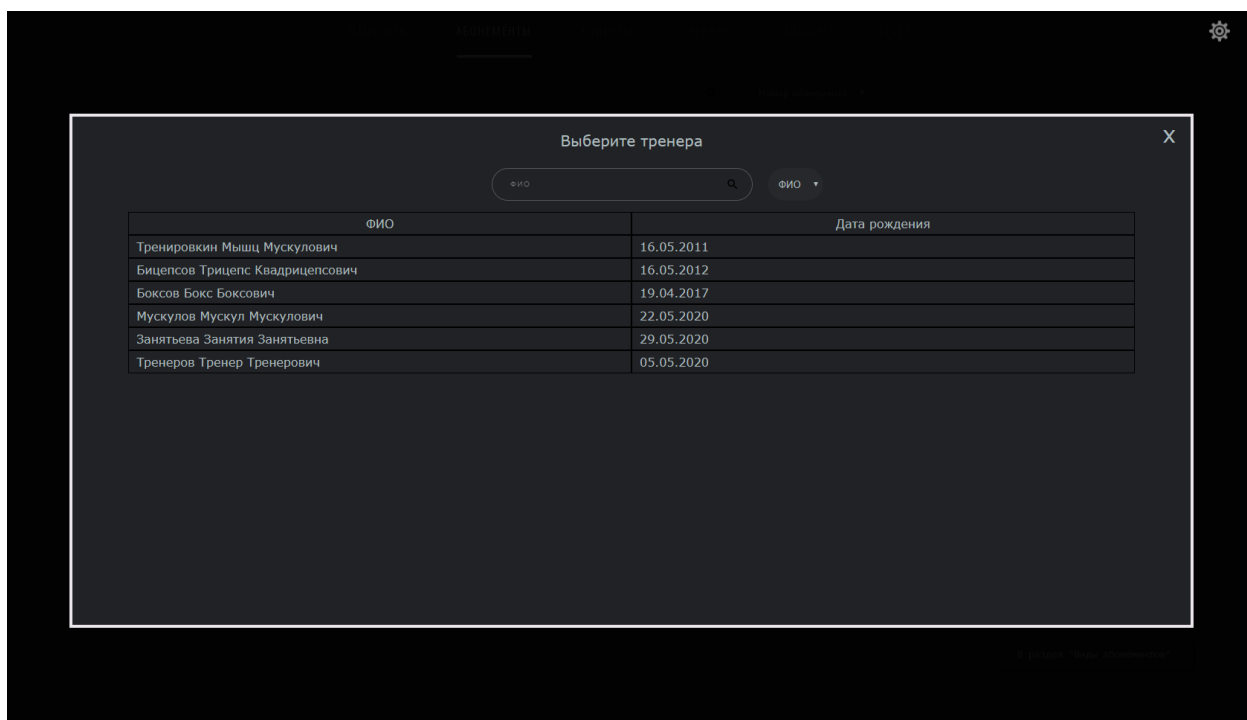


Рисунок 41 – Выбор тренера для добавления абонеента

Рисунок 42 – Заполнение оставшихся полей для добавления абонемента

*Результат: пройден.*

## Тест 12. Редактирование абонемента

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись об абонементе.

*Шаги:*

- перейти в окно «Абонементы»;
- выбрать абонемент из списка;
- нажать кнопку «Изменить»;
- редактировать поля;
- нажать кнопку «Применить».

*Ожидаемый результат:*

- информация об абонементе обновлена, что представлено на рисунке 43.

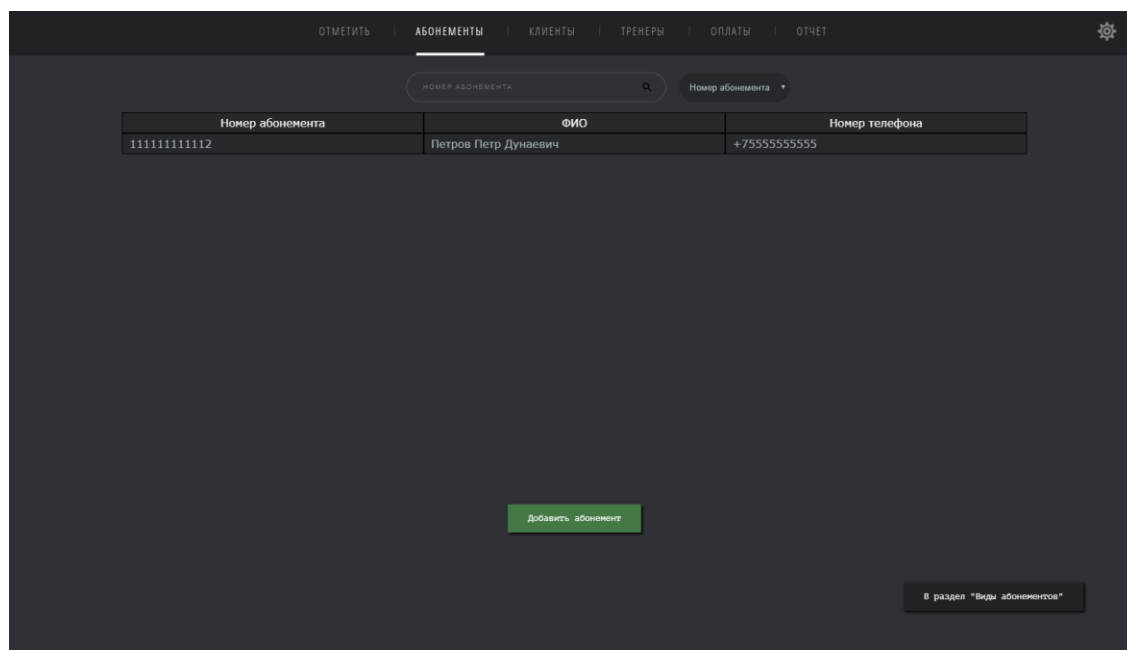


Рисунок 43 – Результат редактирования абонемента

*Результат: пройден.*

**Тест 13.** Удаление абонемента;

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись об абонементе.

*Шаги:*

- перейти в окно «Абонементы»;
- выбрать абонемент из списка;
- нажать кнопку «Удалить»;
- подтвердить выбор;

*Ожидаемый результат:*

- информация об абонементе удалена, что представлено на рисунке 44.

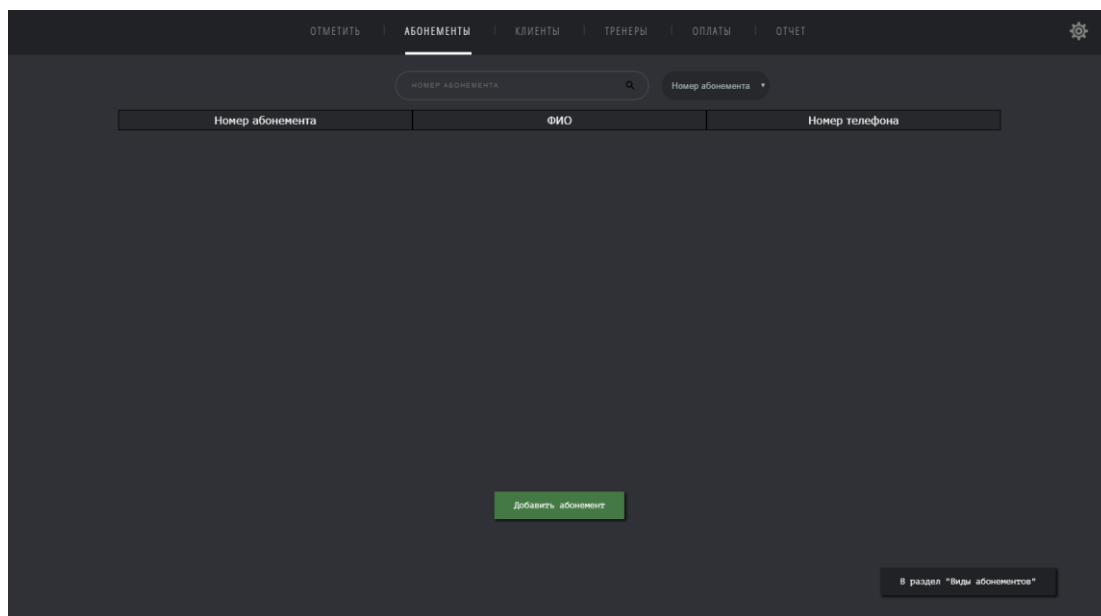


Рисунок 44 – Результат удаления абонемента

*Результат:* **пройден.**

#### **Тест 14. Регистрация посещения**

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись об абонементе.

*Шаги:*

- перейти в окно «Отметить»;
- ввести номер абонемента (действие представлено на рисунке 45);
- нажать кнопку «Отметить»;
- подтвердить выбор;

*Ожидаемый результат:*

- клиент отмечен, количество занятий уменьшено, что представлено на рисунке 46.

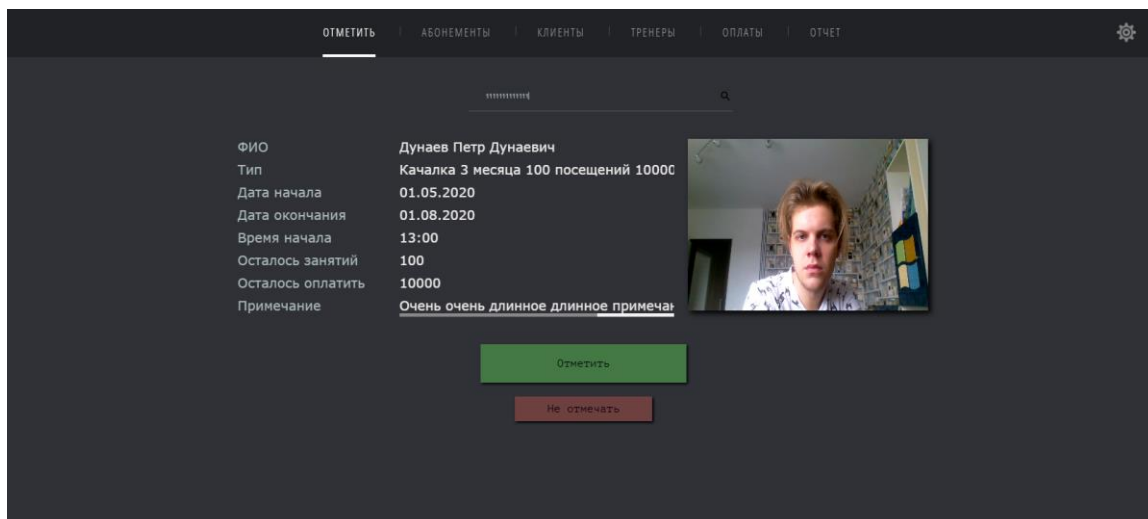


Рисунок 45 – Информация об абонементе в окне «Отметить»

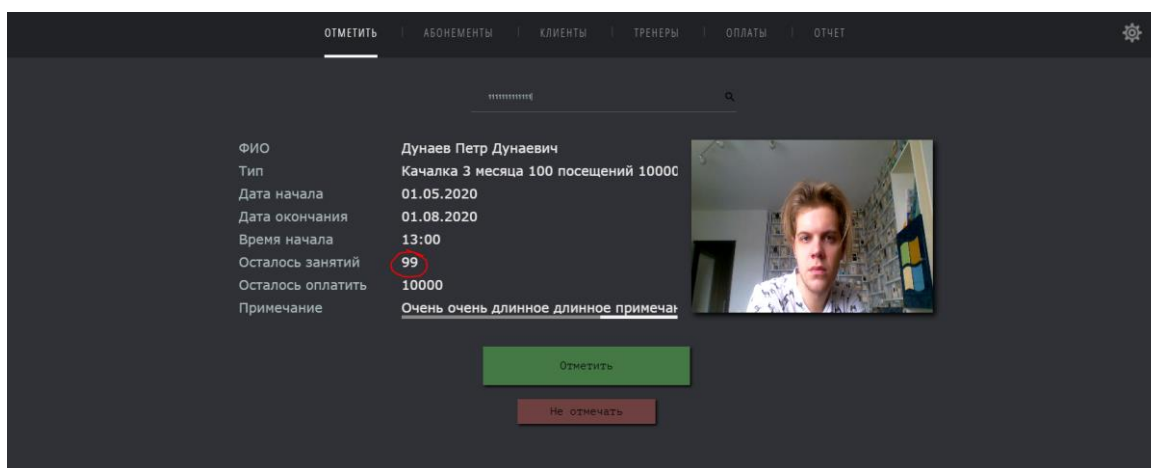


Рисунок 46 – Измененное количество оставшихся занятий.

*Результат: пройден.*

**Тест 15.** Добавление оплаты

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись об абонементе.

*Шаги:*



- перейти в окно «Оплаты»;
- нажать кнопку «Добавить оплату»;
- выбрать абонемент из списка (действие представлено на рисунке 47);
- заполнить оставшиеся поля (действие представлено на рисунке 48);
- нажать кнопку «Добавить».

*Ожидаемый результат:*

- оплата добавлена.

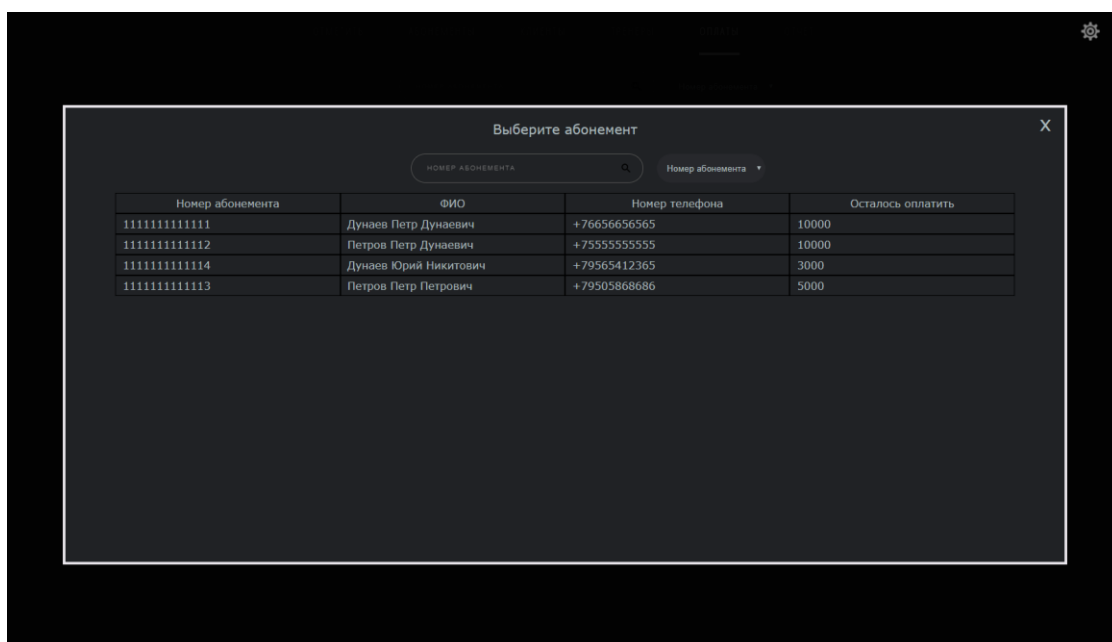


Рисунок 47 – Выбор абонента для добавления оплаты

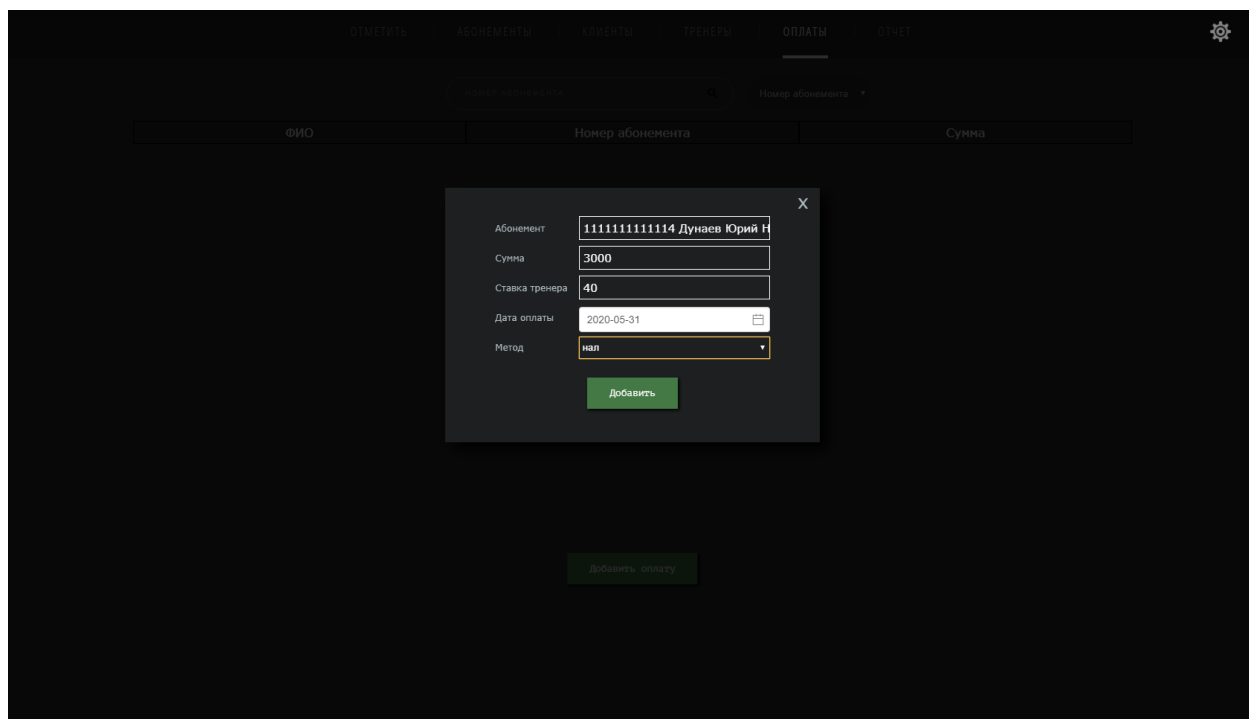


Рисунок 48 – Заполнение полей оплаты

*Результат: пройден.*

#### Тест 16. Удаление оплаты

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись об оплате.

*Шаги:*

- перейти в окно «Оплаты»;
- выбрать оплату из списка;
- нажать на кнопку «Удалить»;
- подтвердить выбор.

*Ожидаемый результат:*

- оплата удалена, что представлено на рисунке 49.

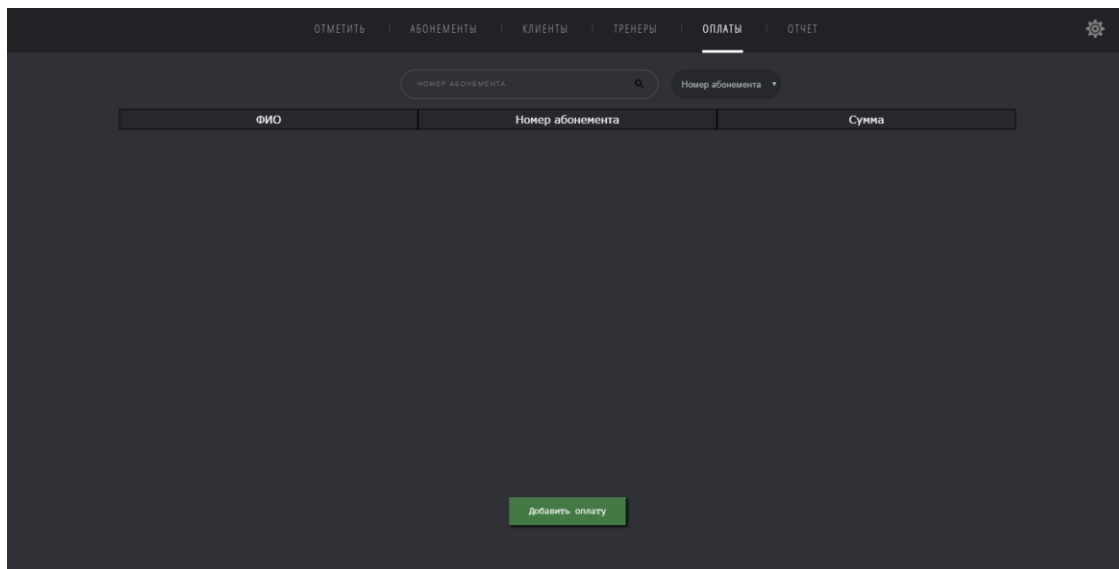


Рисунок 49 – Удаление оплаты

*Результат:* пройден.

**Тест 17.** Просмотр отчета посещений.

*Предусловия:*

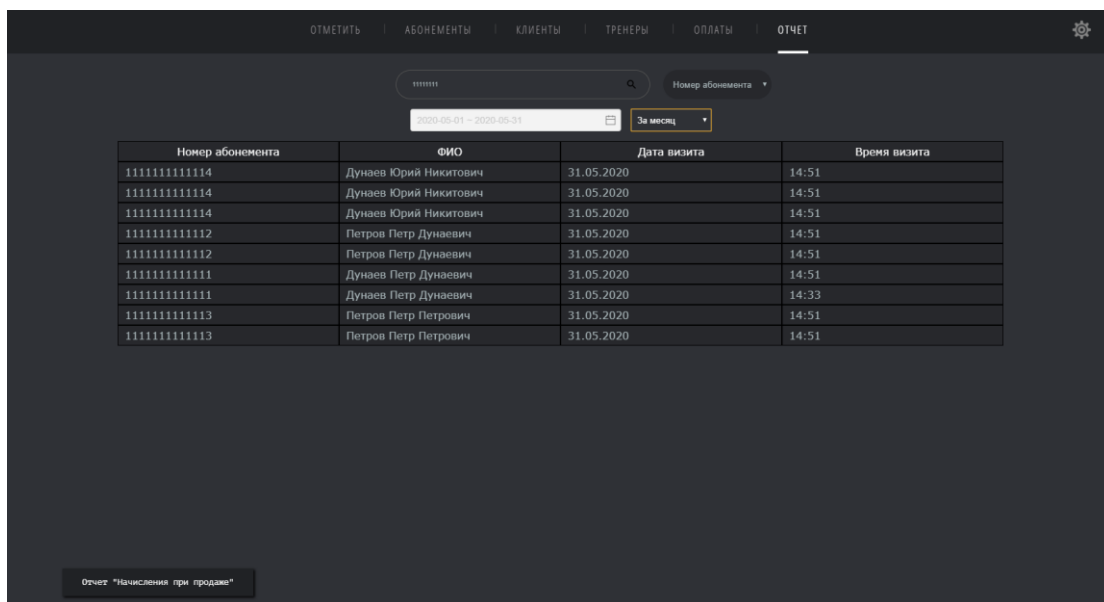
- пользователь должен быть авторизован;
- должен быть отмечен визит клиента.

*Шаги:*

- перейти в окно «Отчет»;
- по желанию заполнить поля фильтра.

*Ожидаемый результат:*

Список посещений показан на рисунке 50.



### Рисунок 50 – Список посещений

**Результат: пройден.**

### Тест 18. Просмотр отчета «Начисления при продаже»

*Предусловия:*

- пользователь должен быть авторизован;
- должна быть запись об оплате.

*Шаги:*

- перейти в окно «Отчет»;
- нажать на кнопку «Отчет «Начисления при продаже»»;
- заполнить поля фильтров.

*Ожидаемый результат:*

- отчет показан на рисунке 51.

ОТМЕТИТЬ   АБОНЕМЕНТЫ   КЛИЕНТЫ   ТРЕНЕРЫ   ОПЛАТЫ   ОТЧЕТ							
Тренер <input type="text" value="Боксов Бокс Боксович"/> Диапазон дат <input type="text" value="2020-04-10 - 2020-06-26"/> <input type="button" value="📅"/> Время начала <input type="text" value="12:00"/> <input type="button" value="x"/>							
Тренер	Номер абонемента	Вид абонемента	Дата оплаты	Сумма оплаты	Ставка тренера	Начисление тренеру	Время начала
Боксов Бокс Боксович	11111111111114	Бокс 30 посещений 3000 рублей	31.05.2020	2000	50	1000	12:00
Боксов Бокс Боксович	11111111111114	Бокс 30 посещений 3000 рублей	31.05.2020	3000	50	1500	12:00
Боксов Бокс Боксович	11111111111113	Бокс 15 посещений 5000 рублей	31.05.2020	5000	50	2500	12:00
<div> <div>Экспорт в Excel</div> <div> Сумма по оплатам: 10000  Сумма по начислениям тренеру: 5000 </div> </div>							

Рисунок 51 – Отчет «Начисления при продаже»

*Результат:* **пройден.**

## 9 Внедрение

Перед внедрением система прошла этап апробации, который происходил в самом крупном зале сети, находящемся по адресу: г. Новосибирск, ул. Зыряновская, 63. По сравнению с другими филиалами, данный зал имеет наибольший поток клиентов и предоставляет самый широкий спектр услуг. Такие показатели позволят понять, как на практике новая система учета будет справляться с большим количеством данных.

Допускалось, что на стадии апробации в системе могут возникнуть ошибки, однако за счет протоколирования изменений в базе данных с помощью триггеров, новая система учета способна вернуться в любое из состояний путем последовательного отката транзакций. Таким образом, в процессе апробации имелась возможность устранить возникающие ошибки и их последствия в кратчайшие сроки без ущерба предприятию.

Перед началом работы все данные из старой базы данных были перенесены в новую. В течении месяца администратор работал с новой системой учета. Система успешно прошла этап апробации и была внедрена в остальные филиалы, что подтверждает Акт внедрения, представленный в приложении Б.

## Заключение

В рамках выпускной квалификационной работы бакалавра было разработано и внедрено приложение для автоматизации учета клиентопотока и движения денежных средств в сети спортивных центров «CHAMPION GYM»

Новая система учета предназначена для автоматизации деятельности администратора спортивного центра. Приложение предоставляет возможность просмотра сетевого отчета для анализа продуктивности отдельных тренеров и эффективности предприятия в целом, а также дает возможность клиентам владеть сетевым абонементом.

В ходе работы были выполнены следующие задачи:

- проведен анализ текущей системы учета, выявлены преимущества и недостатки;
- сформированы требования к новой системе учета и обоснован выбор технологий;
- спроектирована и разработана база данных для системы учета;
- спроектированы и разработаны серверная и клиентская логики приложения;
- проведено функциональное тестирование desktop-приложения;
- произведена апробация и внедрение программного продукта.

В заключение стоит отметить, что поставленная цель достигнута, задачи выполнены в полном объеме. Программный продукт внедрен, что подтверждает Акт внедрения, представленный в приложении Б. Desktop-приложение полностью реализует заявленный функционал и удовлетворяет требованиям удобства, быстродействия и безопасности.

## Список используемых источников

1. Спортивные центры Чемпион [Электронный ресурс] Режим доступа: <https://champion-nsk.com/> (Дата обращения: 13.04.2020).
2. OpenJS Foundation [Электронный ресурс] Режим доступа: <https://nodejs.org/en/> (Дата обращения: 13.04.2020).
3. Express проект Фонда Node.js. [Электронный ресурс] Режим доступа: <https://expressjs.com/ru/> (Дата обращения: 13.04.2020).
4. The PostgreSQL Global Development Group [Электронный ресурс] Режим доступа: <https://www.postgresql.org/> (Дата обращения: 13.04.2020).
5. Knex.js – A SQL Query builder for javascript [Электронный ресурс] Режим доступа: <http://knexjs.org/> (Дата обращения: 13.04.2020).
6. The Progressive JavaScript Framework [Электронный ресурс] Режим доступа: <https://vuejs.org/> (Дата обращения: 13.04.2020).
7. Документация Electron [Электронный ресурс] Режим доступа: <https://www.electronjs.org/docs/tutorial/first-app> (Дата обращения: 13.04.2020).
8. JavaScript-библиотека Electron-Vue [Электронный ресурс] Режим доступа: <https://github.com/SimulatedGREG/electron-vue> (Дата обращения: 19.05.2020).
9. Универсальная система учета «USU» [Электронный ресурс] Режим доступа: <http://usu.kz/> (Дата обращения: 02.04.2020).
10. Qt | Cross Platform software development [Электронный ресурс] Режим доступа: <https://www.qt.io/> (Дата обращения: 02.04.2020).
11. Java Spring Framework [Электронный ресурс] Режим доступа: <https://spring.io/> (Дата обращения: 02.04.2020).



12. JavaFX 14 Documentation [Электронный ресурс] Режим доступа: <https://openjfx.io/> (Дата обращения: 02.04.2020).
13. .NET | Free Cross-Platform [Электронный ресурс] Режим доступа: <https://dotnet.microsoft.com/> (Дата обращения: 02.04.2020).
14. The Web Framework | Django [Электронный ресурс] Режим доступа: <https://www.djangoproject.com/> (Дата обращения: 02.04.2020).
15. PHP: Hypertext Preprocessor [Электронный ресурс] Режим доступа: <https://www.php.net/> (Дата обращения: 02.04.2020).

## Приложение А

### Реализация базы данных

```
CREATE DATABASE sport;

DROP TABLE types;
DROP TABLE subs;
DROP TABLE clients;
DROP TABLE payments;
DROP TABLE trainers;

CREATE TABLE types (
    id SERIAL PRIMARY KEY,
    title varchar(50),
    cost integer,
    training smallint
);

CREATE TABLE subs (
    id SERIAL PRIMARY KEY,
    sub_number varchar(13) UNIQUE,
    sub_status boolean,
    type_id integer,
    client_id integer,
    trainer_id integer,
    left_to_pay integer,
    begin_date date,
    end_date date,
    training_left smallint,
    start_time time,
    note varchar(200)
);

CREATE TABLE clients (
    id SERIAL PRIMARY KEY,
    fio varchar(100),
    phone_number varchar(18),
    first_visit_date date,
    how_to_find varchar(16),
    inviter_id integer,
    note varchar(200)
);

CREATE TABLE payments (
    id SERIAL PRIMARY KEY,
    sub_id integer,
    payment_date date,
    payment_amount integer,
    payment_method varchar(3),
    interest_rate integer
);

CREATE TABLE trainers (
```

```

        id SERIAL PRIMARY KEY,
        fio varchar(100),
        date_birth date
    );

CREATE TABLE visits (
    id SERIAL PRIMARY KEY,
    sub_id integer,
    visit_date date,
    visit_time time
);

CREATE TABLE users (
    username varchar(20),
    userpass varchar(30)
);

-- Ограничения на таблицы
ALTER TABLE subs ADD CHECK (type_id > 0);
ALTER TABLE subs ADD CHECK (client_id > 0);
ALTER TABLE subs ADD CHECK (trainer_id > 0);
ALTER TABLE subs ADD CHECK (begin_date < end_date);

ALTER TABLE subs ADD CONSTRAINT membershipfk FOREIGN KEY (client_id)
REFERENCES clients (id) ON UPDATE NO ACTION ON DELETE CASCADE;

ALTER TABLE subs ADD CONSTRAINT trainerfk FOREIGN KEY (trainer_id)
REFERENCES trainers (id) ON UPDATE NO ACTION ON DELETE SET NULL;

ALTER TABLE subs ADD CONSTRAINT typefk FOREIGN KEY (type_id)
REFERENCES types (id) ON UPDATE NO ACTION ON DELETE SET NULL;

-- Триггер для протоколирования изменений 'types'
CREATE TABLE types_audit (
    operation varchar(1) NOT NULL,
    stamp timestamp NOT NULL,
    id integer,
    title varchar(50),
    cost integer,
    training smallint
);

CREATE FUNCTION process_types_audit() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO types_audit SELECT 'D', now(), OLD.*;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO types_audit SELECT 'U', now(), NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO types_audit SELECT 'I', now(), NEW.*;
        RETURN NEW;
    END IF;
    RETURN NULL;

```

```

    END;
$$LANGUAGE plpgsql;

CREATE TRIGGER types_audit
AFTER INSERT OR UPDATE OR DELETE ON types
    FOR EACH ROW EXECUTE PROCEDURE process_types_audit();

-- Триггер для протоколирования изменений 'subs'
CREATE TABLE subs_audit (
    operation varchar(1) NOT NULL,
    stamp timestamp NOT NULL,
    id integer,
    sub_number varchar(13),
    sub_status boolean,
    type_id integer,
    client_id integer,
    trainer_id integer,
    left_to_pay integer,
    begin_date date,
    end_date date,
    training_left smallint,
    start_time time,
    note varchar(200)
);

CREATE FUNCTION process_subs_audit() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO subs_audit SELECT 'D', now(), OLD.*;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO subs_audit SELECT 'U', now(), NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO subs_audit SELECT 'I', now(), NEW.*;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER subs_audit
AFTER INSERT OR UPDATE OR DELETE ON subs
    FOR EACH ROW EXECUTE PROCEDURE process_subs_audit();

-- Триггер для протоколирования изменений 'clients'
CREATE TABLE clients_audit (
    operation varchar(1) NOT NULL,
    stamp timestamp NOT NULL,
    id integer,
    fio varchar(100),
    phone_number varchar(18),
    first_visit_date date,
    how_to_find varchar(16),
    inviter_id varchar(30),

```

```

        note varchar(200)
    );

CREATE FUNCTION process_clients_audit() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO clients_audit SELECT 'D', now(), OLD.*;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO clients_audit SELECT 'U', now(), NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO clients_audit SELECT 'I', now(), NEW.*;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER clients_audit
AFTER INSERT OR UPDATE OR DELETE ON clients
FOR EACH ROW EXECUTE PROCEDURE process_clients_audit();

-- Триггер для протоколирования изменений 'payments'
CREATE TABLE payments_audit (
    operation varchar(1) NOT NULL,
    stamp timestamp NOT NULL,
    id integer,
    sub_id integer,
    payment_date date,
    payment_amount integer,
    payment_method varchar(3),
    interest_rate integer
);

CREATE FUNCTION process_payments_audit() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO payments_audit SELECT 'D', now(), OLD.*;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO payments_audit SELECT 'U', now(), NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO payments_audit SELECT 'I', now(), NEW.*;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER payments_audit
AFTER INSERT OR UPDATE OR DELETE ON payments
FOR EACH ROW EXECUTE PROCEDURE process_payments_audit();

```

```

-- Триггер для протоколирования изменений 'trainers'
CREATE TABLE trainers_audit (
    operation varchar(1) NOT NULL,
    stamp timestamp NOT NULL,
    id integer,
    fio varchar(100),
    date_birth date
);

CREATE FUNCTION process_trainers_audit() RETURNS TRIGGER AS $$
BEGIN
    IF (TG_OP = 'DELETE') THEN
        INSERT INTO trainers_audit SELECT 'D', now(), OLD.*;
        RETURN OLD;
    ELSIF (TG_OP = 'UPDATE') THEN
        INSERT INTO trainers_audit SELECT 'U', now(), NEW.*;
        RETURN NEW;
    ELSIF (TG_OP = 'INSERT') THEN
        INSERT INTO trainers_audit SELECT 'I', now(), NEW.*;
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$$LANGUAGE plpgsql;

CREATE TRIGGER trainers_audit
AFTER INSERT OR UPDATE OR DELETE ON trainers
FOR EACH ROW EXECUTE PROCEDURE process_trainers_audit();

```

## Приложение Б.

Общество с ограниченной  
ответственностью

**«Чемпион»**

Тел.: 248-80-48

E-mail: champion-nsk@mail.ru

Сайт: champion-nsk.com



Адрес: 630102, г. Новосибирск, ул. Зыряновская, 63

ИНН/КПП 5405954156/540501001

ОГРН 1155476026529

Р/с 40702810420300100033

Ф-л Новосибирский №2 ПАО Банк «ФК Открытие»  
630004 г. Новосибирск, ул. Ленина, д. 18.

К/с 30101810350040000741

ИНН/БИК 7706092528/045004741

КПП 540743001

### АКТ ВНЕДРЕНИЯ

Настоящий акт подтверждает, что результат выпускной квалификационной работы студента НГТУ группы АВТ-610 очной формы обучения Дунаева Н. Ю., на тему «Разработка Desktop-приложения для автоматизации учёта клиентопотока и движения денежных средств в сети спортивных центров» внедрен в деятельность компании и используются для учета клиентов.

Благодаря внедрению разработанной студентом системы учета, удалось:

- 1) Уменьшить количество операционных действий, регулярно совершаемых администратором за счет более удобной реализации базового функционала
- 2) Облегчить процесс подготовки сотрудников для работы с системой учета вследствие упрощения интерфейса
- 3) Обеспечить сохранность данных при помощи разграничения прав доступа

Генеральный директор  
ООО «Чемпион»



Дунаев Н. Ю.