МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Факультет информационных технологий и компьютерной безопасности  
Кафедра компьютерных и интеллектуальных технологий проектирования

КУРСОВОЙ ПРОЕКТ

по дисциплине «Анализ данных и машинное обучение»

Тема «Визуальный анализ данных и построение моделей машинного обучения»

Расчётно-пояснительная записка

Разработал студент Талдыкин И.С.

Подпись, дата Инициалы, фамилия

Руководитель Гусев П.Ю.

Подпись, дата Инициалы, фамилия

Члены комиссии

Подпись, дата Инициалы, фамилия

Подпись, дата Инициалы, фамилия

Нормоконтролер Гусев П.Ю.

Подпись, дата Инициалы, фамилия

Защищена Оценка

дата

Воронеж 2022 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ  
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(ФГБОУ ВО «ВГТУ», ВГТУ)

Кафедра компьютерных и интеллектуальных технологий проектирования

ЗАДАНИЕ  
на курсовой проект

по дисциплине «Анализ данных и машинное обучение»

Тема «Визуальный анализ данных и построение моделей машинного обучения»

Студент группы бПО-191 Талдыкин Игорь Сергеевич

Фамилия, имя, отчество

Номер варианта 17

Технические условия MS Visual Studio Code, Python, Pandas, Numpy, Seaborn, SKlearn.

Содержание и объем работы (графические работы, расчеты и прочее): произвести визуальный анализ данных и построить модели машинного обучения на основе выборки, 47 страниц, 1 таблица, 38 рисунков и 13 листингов.

Сроки выполнения этапов Февраль – Май 2022

Срок защиты курсовой работы

Руководитель Гусев П.Ю.

Подпись, дата Инициалы, фамилия

Задание принял студент Талдыкин И.С.

Подпись, дата Инициалы, фамилия

Замечания руководителя

Содержание

[Введение 5](#_Toc73402208)

[1 Постановка задачи 6](#_Toc73402209)

[1.1 Концептуальное проектирование 6](#_Toc73402210)

[1.2 Логическое проектирование 8](#_Toc73402211)

[1.3 Физическое проектирование 8](#_Toc73402212)

[1.3.1 Выбор СУБД 8](#_Toc73402213)

[1.3.2 Физическая модель базы данных 8](#_Toc73402214)

[2 Реализация железнодорожной информационно-справочной системы 10](#_Toc73402215)

[2.1 Реализация базы данных 11](#_Toc73402216)

[2.2 Реализация приложения 15](#_Toc73402217)

[2.2.1 Подключение базы данных 15](#_Toc73402218)

[2.2.2 Реализация таблиц 17](#_Toc73402219)

[2.2.3 Вкладка «Личный кабинет» 19](#_Toc73402220)

[2.2.4 Вкладка «Информация» 20](#_Toc73402221)

[2.2.5 Локализация приложения 22](#_Toc73402222)

[3 Тестирование железнодорожной информационно-справочной системы 30](#_Toc73402223)

[Заключение 40](#_Toc73402224)

[Список литературы 41](#_Toc73402225)

[Приложение А 42](#_Toc73402226)

[Приложение Б 44](#_Toc73402227)

# Введение

Табличное представление информации – одно из самых эффективных представлений для человеческого восприятия. Именно в виде таблиц хранится большинство информации в цифровом виде, это помогает сильно ускорить поиск нужных данных и с технической стороны.

Одной из повсеместно используемых технологий являются «Базы данных», они широко распространены во всех сферах жизни: ведение бухгалтерской отчетности, учёт при посадке урожая в аграрной промышленности, учёт студентов, сотрудников и школьников в соответствующих заведениях.

Базы данных представляют собой некоторый набор постоянно хранимых данных, систематизированных таким образом, чтобы эти данные могли быть найдены и обработаны вычислительной машиной с помощью соответствующего прикладного программного обеспечения. Такое программное обеспечение называется системой управления базами данных (СУБД). Данные обычно хранятся в виде строк и столбцов формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов «SQL».

Целью курсового проекта является создания CRUD-приложения реализующую базу данных. А так же получение навыка по созданию таких приложений, и навыков по проектировании баз данных.

# 1 Постановка задачи

Требуется разработать базу данных для создания железнодорожной информационно-справочной системы. Пользователь может покупать билеты на созданные заранее маршруты, на деньги которые хранятся на его личном балансе. Администратор составляет маршруты для пользователей из таких сущностей как «Город», «Маршрут» и «поезд» (состоящий из двух сущностей «Локомотив» и «Вагон»).

## 1.1 Концептуальное проектирование

Для начала была построена концептуальная модель базы данных. Данная модель представляет собой описание предметной области, для которой разрабатывается база данных. На данном этапе были определены такие сущности как «Город», «Путь», «Маршрут», «Поезд», «Локомотив», «Вагон», «Билет» и «Пользователь». Модель не отвечает за физическую реализацию. [4] Данный концепт представлен на рисунке 1.

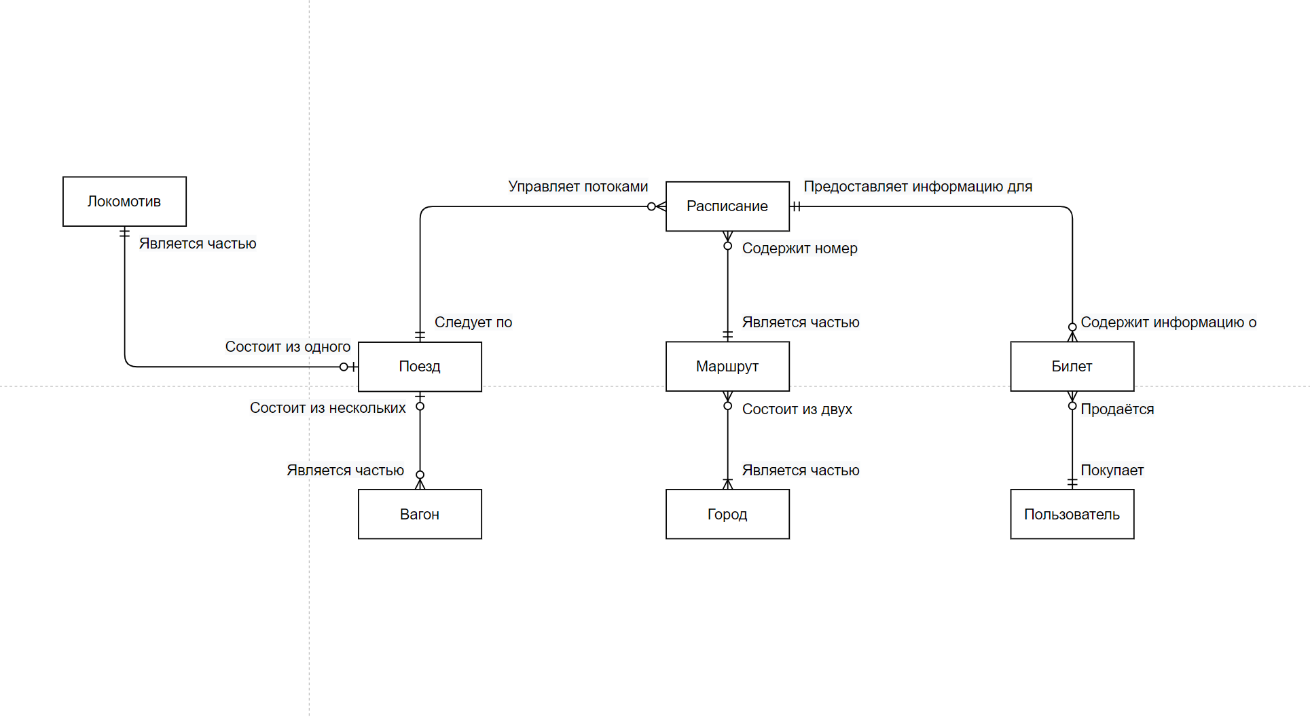


Рисунок 1 – Концептуальная модель базы данных

Также на данном этапе была построена модель использования. Данная модель представляет собой варианты использования приложения разными типами пользователей:

- admin;

- customer.

Данная модель представлена на рисунке 2.

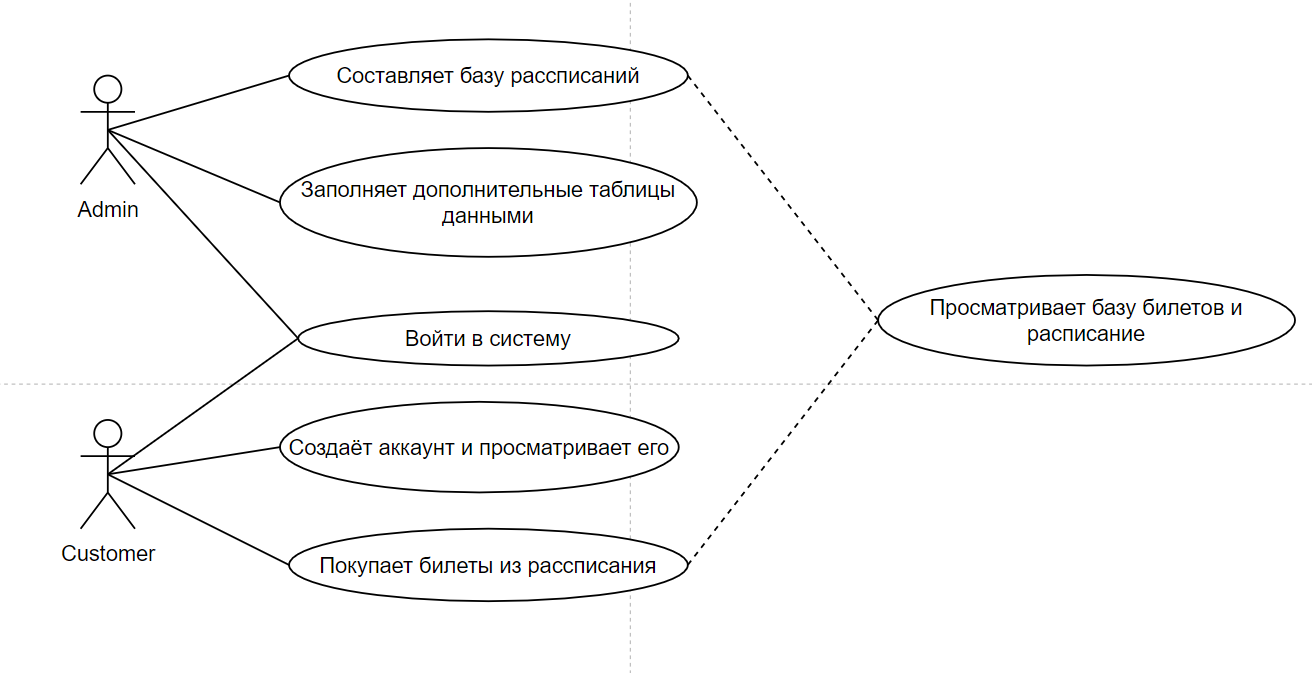


Рисунок 2 – Модель использования

## 1.2 Логическое проектирование

Далее была создана логическая модель базы данных. Она определяет атрибуты сущностей БД. Данная модель представлена на рисунке 3.

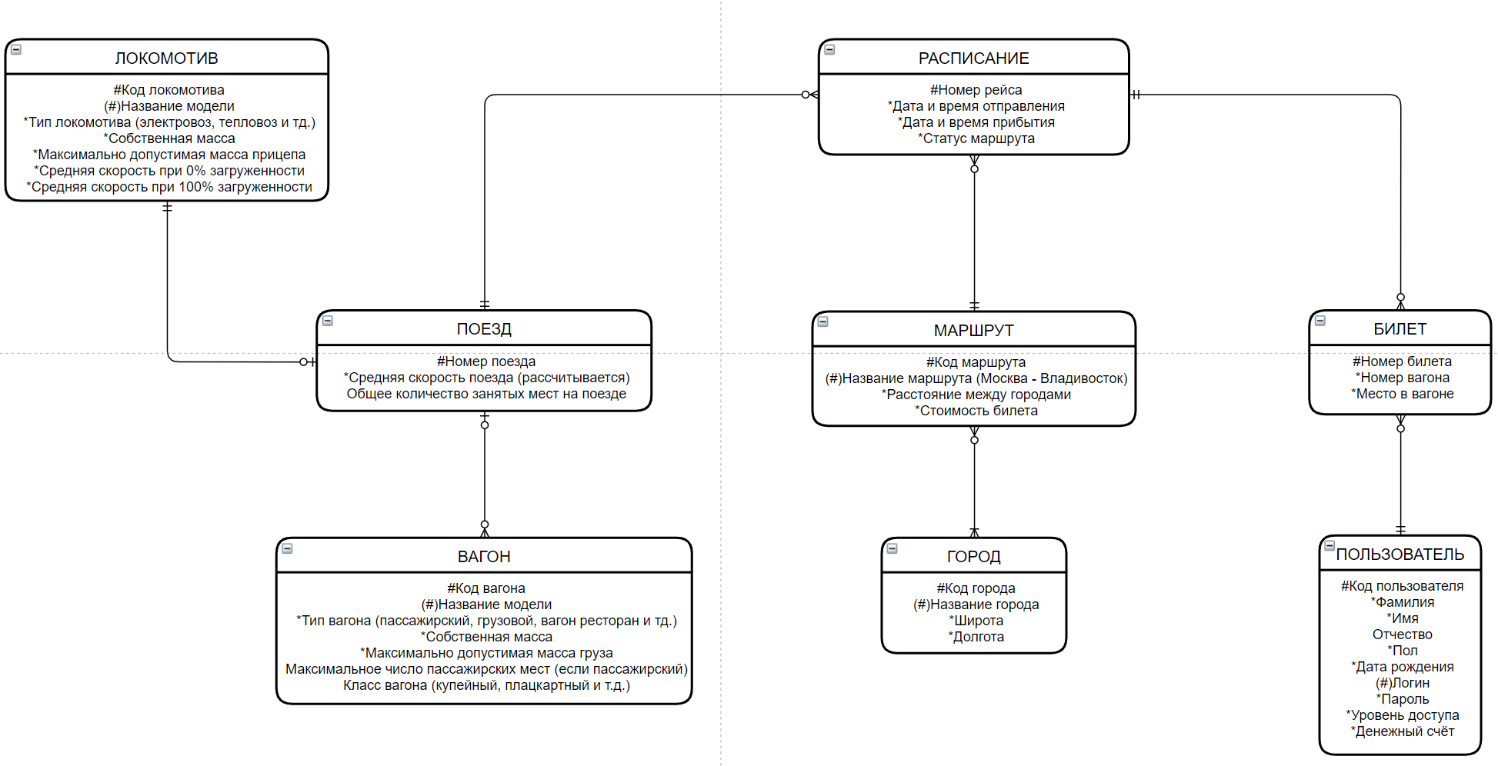


Рисунок 3 – Логическая модель базы данных

## 1.3 Физическое проектирование

### 1.3.1 Выбор СУБД

На этом этапе проектирования была выбрана СУБД – ею стала «SQLite». Данный выбор обоснован удобством переноса базы данных между разными устройствами. С использованием SQLite размер файла БД значительно уменьшается. SQLite также оснащён графической оболочкой СУБД – «SQLite Data Base Browser».

### 1.3.2 Физическая модель базы данных

При проектировании конечной стадии базы данных была создана её физическая модель. Она определяет типы данных для каждого атрибута в таблице, таким образом, например, такие атрибуты как Имя и Фамилия получают используют тип данных «nvarchar» длинной в 20 символов, данное решение обусловлено тем, что почти у всех людей в мире имя и фамилия не превышают длины в 20 букв латинского алфавита. Так же в физической модели были назначены родительские и внешние ключи (PK и FK соответственно), сохраняя реализуя тем самым логические связи сущностей. Физическая модель базы данных представлена на рисунке 4.

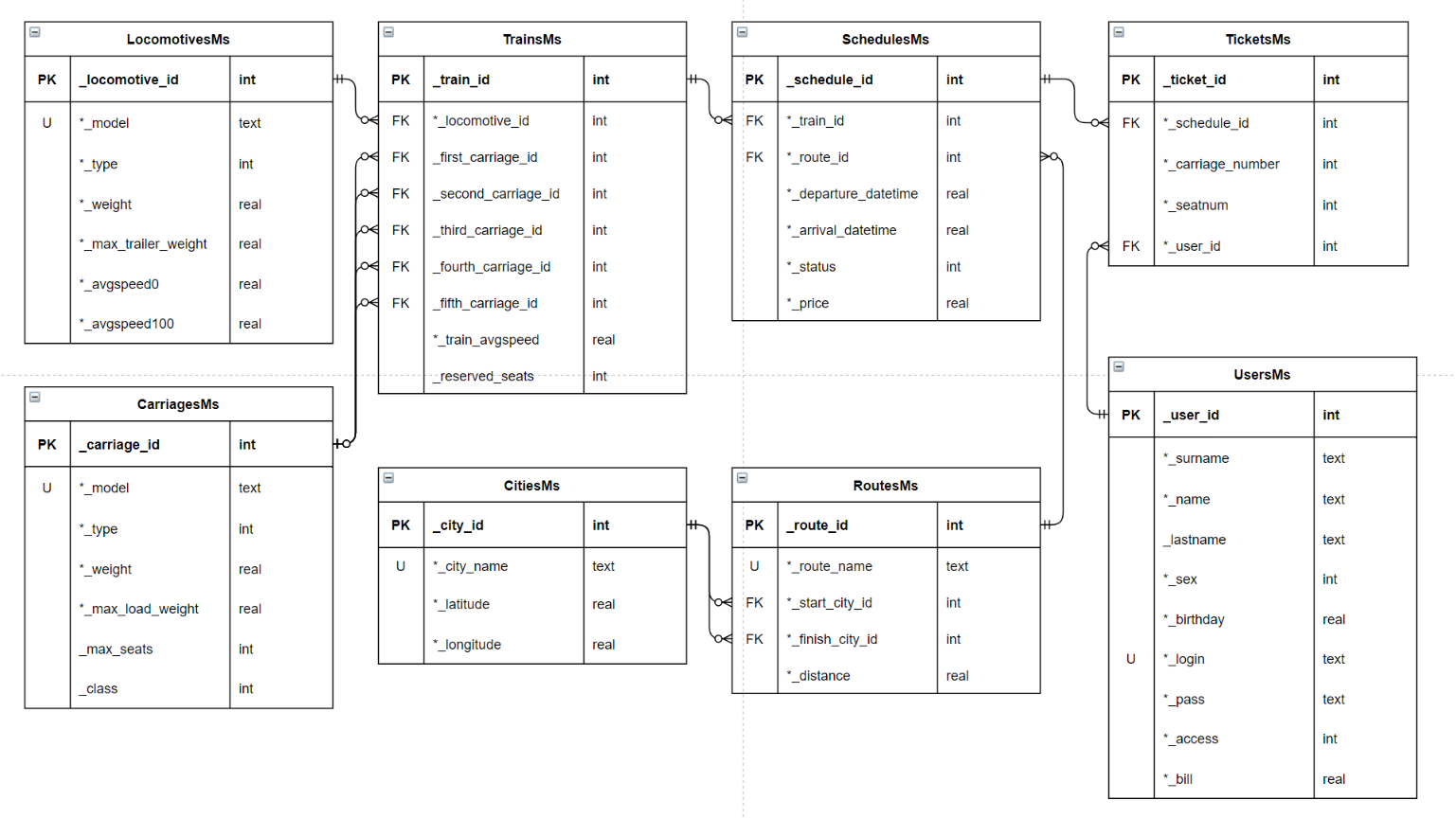


Рисунок 4 – Физическая модель базы данных

# 2 Реализация железнодорожной информационно-справочной системы

При реализации базы данных были поделены обязанности со вторым участником проекта – Никулиным Владимиром. В таблице 1 представлены поделенные обязанности для реализации проекта.

Таблица 1 – Распределение обязанностей

|  |  |  |
| --- | --- | --- |
|  | Талдыкин И.С. | Никулин В.С. |
| Таблицы базы данных | «Расписание», «Локомотивы», «Вагоны», «Поезда» | «Города», «Маршруты», «Билеты», «Пользователи». |
| Реализация базы данных (код приложения) | Подключение базы данных к проекту. Реализация таблиц «Расписание», «Локомотивы», «Вагоны», «Поезда». Реализация покупки билетов пользователем. Реализация запроса для таблицы «Расписание». | Реализация таблицы «Города», «Маршруты», «Билеты», «Пользователи». Реализация паттерна MVVM для архитектуры приложения. Реализация моделей сущностей в виде классов. Реализация запроса для таблицы «Пользователи». |
| Реализация базы данных (дизайн приложения) | Вкладка «Личный кабинет», «Информация». Локализация приложения для 3-х языков. | Окна регистрации/авторизации, вкладка «настройки». Визуальное оформление приложения. Смена тем. |

Репозиторий проекта хранится на удаленном репозитории GitHub, ссылка:

- <https://github.com/JoelWillLiveForever/BataDase/tree/timeCapsule>.

## 2.1 Реализация базы данных

Для начала была создана база данных с помощью СУБД SQLite, название базы данных дано «TrainsDB». После чего в диалоговом окне были добавлены спроектированные в физической модели таблицы. Код SQL-запроса на создание таблицы сразу отображается внизу окна. Коды запросов всех таблицы представлены в Приложении А. Пример добавления таблицы представлен на рисунке 5.

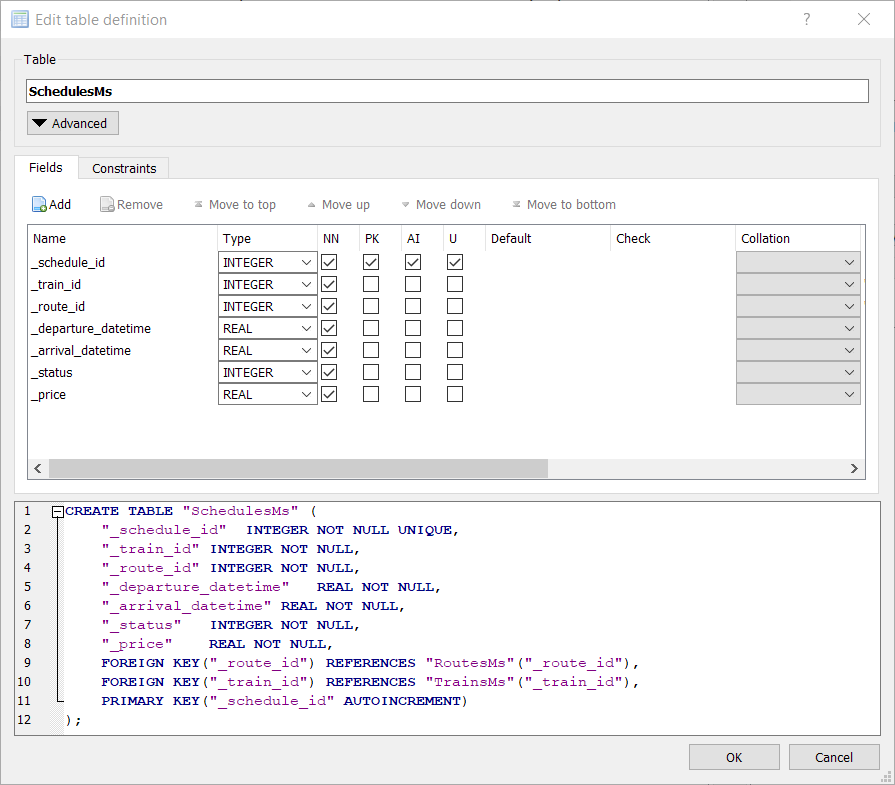


Рисунок 5 – Пример создания таблицы «SchedulesMs»

На рисунке 5 видно, что таблица «SchedulesMs» использует внешние ключи таблиц «TrainsMs» и «RoutesMs» для последующего вывода в таблицу через запрос номера поезда и номера маршрута. Так же сущность расписания имеет атрибуты времени отправки, времени прибытия, статуса и цены. Статус имеет 3 значения – «Активен», «В пути» и «Завершён».

Далее была создана таблица «LocomotivesMs», она хранит в себе атрибуты названия модели локомотива, типа локомотива, имеющего 3 значения – «паровоз», «электровоз» и «тепловоз», массы самого локомотива и максимальной массы прицепляемых вагонов, а так же скорость одиночного локомотива и скорость локомотива с прицепленными вагонами. Создание таблицы представлено на рисунке 6.

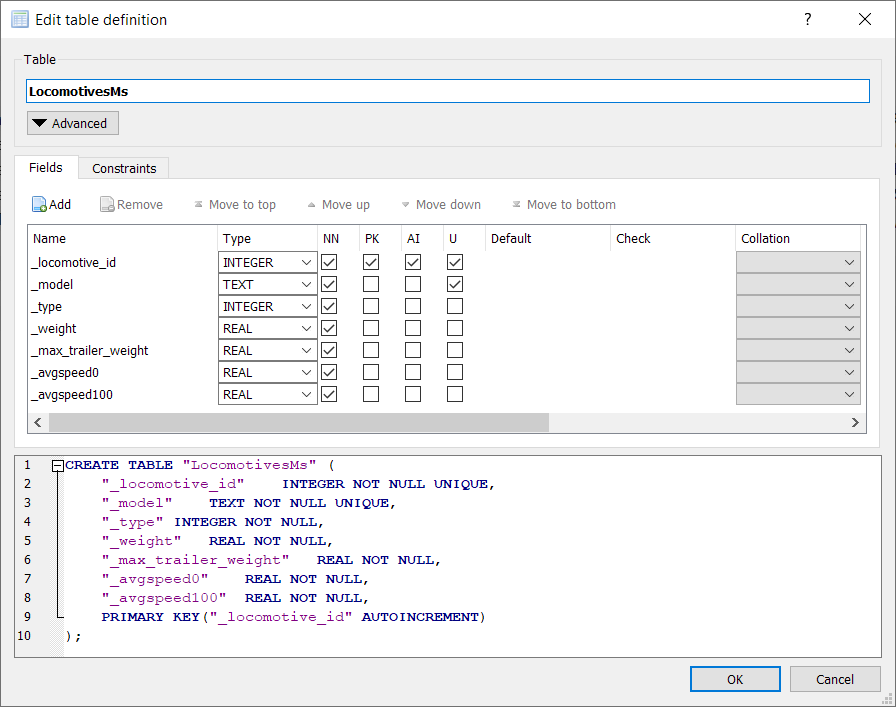


Рисунок 6 – Создание таблицы «LocomotivesMs»

Далее была создана таблица «CarriagesMs», реализующая в себе сущность вагона. По аналогии с «LocomotivesMs» она хранит такие атрибуты как модель, тип, имеющий 3 значения – «пассажирский», «ресторанный» и «грузовой», масса вагона, масса погруженного вагона, количество мест в вагоне и класс имеющий также 3 значения – «купе», «плацкарт» и «сидячий». Создание таблицы представлено на рисунке 7.

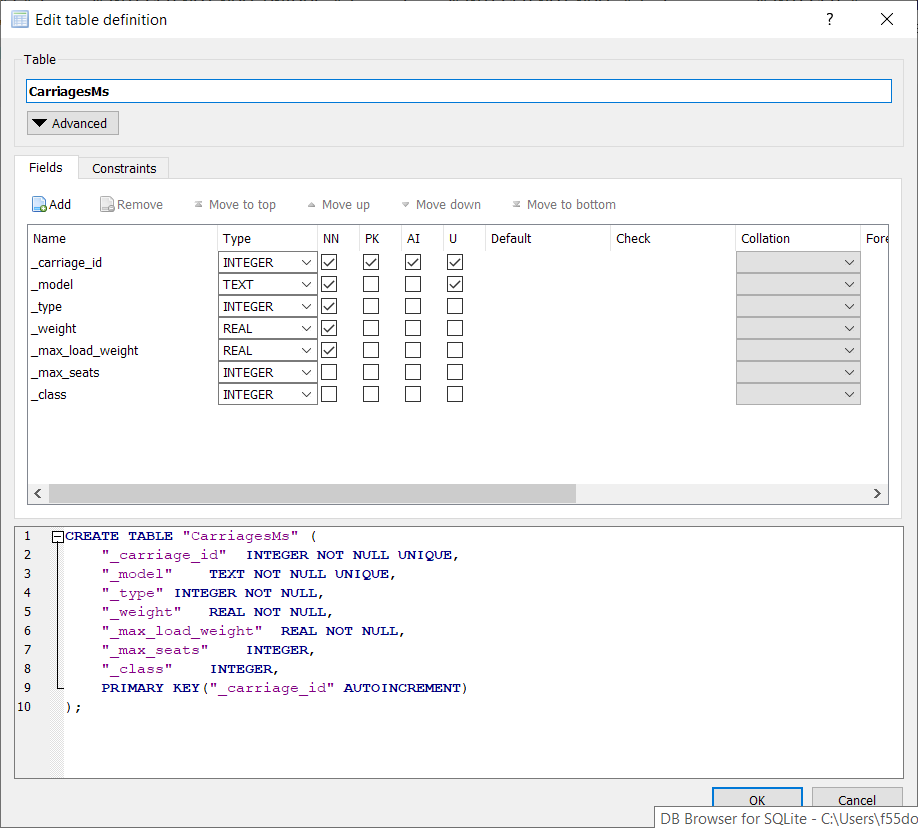


Рисунок 7 – Создание таблицы «CarriagesMs»

Далее была создана таблица «TrainsMs», реализующую в себе сущность поезда. Она использует внешние ключи локомотива и 5 ключей вагонов, так как при проектировании базы данных было заявлено условие, что поезд содержит в себе обязательно один локомотив и не более пяти вагонов. Так же таблица использует атрибуты средней скорости поезда и общее количество мест для пассажиров. Создание таблицы представлено на рисунке 8.

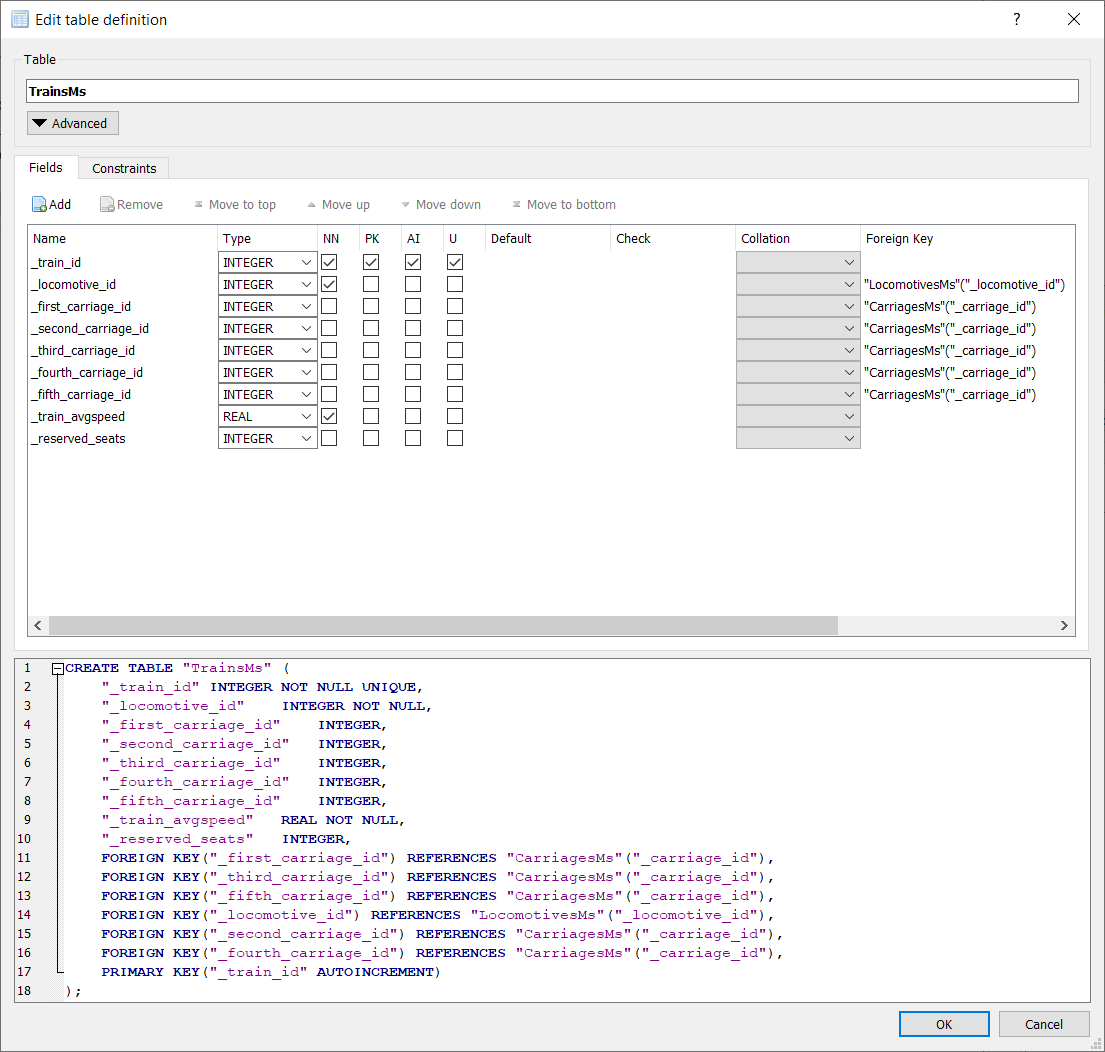


Рисунок 8 – Создание таблицы «TrainsMs»

## 2.2 Реализация приложения

### 2.2.1 Подключение базы данных

Для подключения базы данных к проекту использовались пакеты «Entity.Framework» и «SQLite», подключенные с помощью менеджера пакетов «NuGet». После чего в xaml-файле «App.config» были добавлены строки соединения и провайдера. Код строк показан на листинге 1.

Листинг 1.

<connectionStrings>

<add name="DefaultConnection" connectionString="Data Source=.\Resources\Databases\Trains.db" providerName="System.Data.SQLite" />

</connectionStrings>

<entityFramework>

<defaultConnectionFactory type="System.Data.Entity.Infrastructure.LocalDbConnectionFactory, EntityFramework">

<parameters>

<parameter value="v11.0" />

</parameters>

</defaultConnectionFactory>

<providers>

<provider invariantName="System.Data.SQLite" type="System.Data.SQLite.EF6.SQLiteProviderServices, System.Data.SQLite.EF6" />

<provider invariantName="System.Data.SqlClient" type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />

<provider invariantName="System.Data.SQLite.EF6" type="System.Data.SQLite.EF6.SQLiteProviderServices, System.Data.SQLite.EF6" />

</providers>

</entityFramework>

Завершающим этапом подключения базы данных было создания C# файла «AppDBContext» – контекста базы данных. В нём прописаны связи между таблицами в базе данных, тем самым позволяя им обращаться через внешние ключи друг другу. Код класса представлен на рисунке 9.



Рисунок 9 – Код класса AppDBContext

### 2.2.2 Реализация таблиц

Используя паттерн MVVM, в сотрудничестве с Никулиным Владимиром были, созданы классы ViewModel для всех таблиц. Ниже будет описана лишь половина.

В классе «LocomotivesVM» была реализована обработка, добавление изменение и удаления поступающей информации посредствам получения модели. Листинг данного класса представлен в Приложении Б. Как видно из листинга что в конструкторе класса задаются параметры TextBlock'ов, TextBox'ов и ComboBox'а – это сделано для отображения контекстного диалога, экземпляра класса DialogV. Данные элементы пользовательского интерфейса служат инструментом выбора параметров для заполнения той или иной таблицы. Такой подход реализован и в остальных классах типа VM (ViewModel). В методе «ConnectAndUpdate» реализовано постоянное обновление наполнения таблицы, данный метод реализован аналогично для всех классов. В методе «AddEdit» происходит построение контекстного диалогового окна DialogV, в него добавляются элементы пользовательского интерфейса, в зависимости от параметра «IsAdd» определяется контент кнопки при добавлении или изменении соответственно и кнопке определяется метод действия «ExecuteAddEdit». В методе «ExecuteAddEdit» сначала проводится проверка на правильность введённых пользователем данных, в случае ошибки выводится соответствующее сообщение. Далее проверка уникальности модели того вводимого локомотива – она реализована с помощью SQL-запроса. После чего идёт создание временного объекта модели локомотива, и если это добавление объект добавляется в таблицу с последующим очищением полей ввода, в противном случае с помощью SQL-запроса Single происходит изменение найденного объекта таблицы. В методе «Delete» реализовано удаление элемента таблицы с помощью его выделения, учитывая контекст удаления – то есть удаление происходит и в зависящих таблицах, в данном примере в таблице «TrainsMs», с последующим сохранением контекста базы данных. Данный метод является одинаковым для всех таблиц в программе.

В классе «CarriagesVM» по аналогии с «LocomotivesVM» была реализована обработка, добавление, изменение и удаление поступающей информации посредствам получения модели. Листинг данного класса также представлен в Приложении Б. Отличием от класса локомотивов является изменённый метод «Delete», в нём каждый вагон при контекстном удалении удаляется из таблицы с поездами.

В классе «TrainsVM» по аналогии с ранее созданными классами, была реализована обработка, добавление, изменение и удаление поступающей информации посредствам получения модели. Листинг данного класса также представлен в Приложении Б. Отличием от классов локомотивов и вагонов является использование атрибутов других таблиц при создании объекта «поезд» в методе «ExecuteAddEdit». Данная модификация была реализована с помощью SQL-запроса «FirstOrDefault». Это позволило предотвратить прекращение работы программы из-за исключения о null объекте.

В классе «SchedulesVM» была реализована обработка, добавление, изменение и удаление поступающей информации посредствам получения модели. Листинг данного класса также представлен в Приложении Б. Отличием от всех выше перечисленных классов является разделение на логику для администратора и покупателя. В конструкторе класса и методе «AddEdit» было реализовано добавление полей места в вагоне и номера самого вагона на контекстное диалоговое окно. Вся логика разделяется с помощью условия глобального параметра «IsAdmin», если оно равно false, то работает логика пользователя. Сама по себе логика пользователя подразумевает под собой покупку билетов из расписания. В методе «ExecuteBuy» определяется локальная цена выбранного элемента расписания, определяется количество денег на счету пользователя и сравнивается, если денег не достаточно для покупки, то приложение выдаст соответствующую ошибку, в противном случае, деньги списываются со счёта пользователя декрементируя его. Далее происходит проверка на ввод данных информации о выбранном вагоне и месте в нём, после чего создается новый объект класса «TicketsM», ему присваиваются выше указанные данные, а так же ID текущего пользователя через глобальный параметр «CurrentUserID». После всех действий выводится сообщение об успешной покупке билета, и месте где можно просмотреть информацию о нём.

Так же в классе были реализованы методы параметризованного запроса «Request» и «ExecuteRequest», в первом реализована логика создания контекстного диалогового окна DialogV, а во втором сам запрос который выводит в уже созданную таблицу в графическом интерфейсе, все записи соответствующие введённому пользователем атрибуту название маршрута.

### 2.2.3 Вкладка «Личный кабинет»

Для отображения актуальной информации об аккаунте пользователя, была разработана вкладка «личный кабинет», в ней можно найти такую информацию как Ф.И.О. текущего пользователя, его логин и денежный счёт в рублях. Также в данной вкладке этот счёт можно и пополнить с помощью кнопки «Пополнить счёт», отображается контекстное диалоговое окно с полем ввода суммы, на которую надо пополнить счёт. В коде логика запускается с помощью метода «ExecuteAddMoney», в нём сначала проходи проверка И последним элементом окна является кнопка «Выйти», с её помощью пользователь выходит из своего аккаунта чтобы дать другому зайти в свой. Интерфейс данного функционала представлен на рисунке 10.

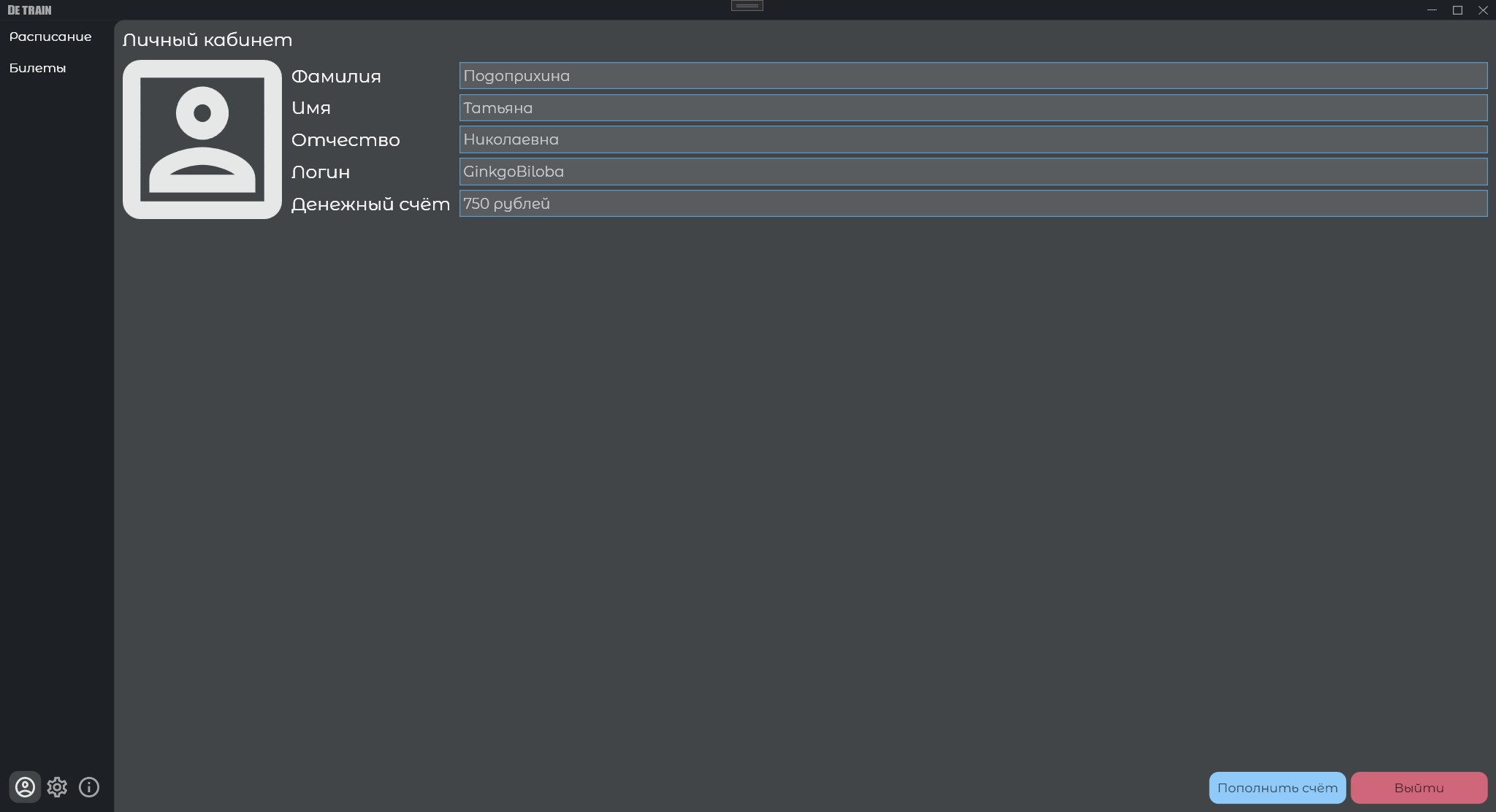


Рисунок 10 – Вкладка «Личный кабинет»

### 2.2.4 Вкладка «Информация»

Для отображения актуальной информации о приложении, была разработана вкладка «информация» позволяющая узнать пользователю суть приложения и информацию о создателях программы, а также такую информацию как версия и билд приложения. Для понятности большему кругу пользователей, приложение включая эту вкладку было локализировано на 3 языках – русский, английский и японский (см. пункт 2.2.5). Интерфейс данного функционала представлен на рисунке 11. Так же переведённый на английский и японский языки интерфейс представлен на рисунках 12 и 13.

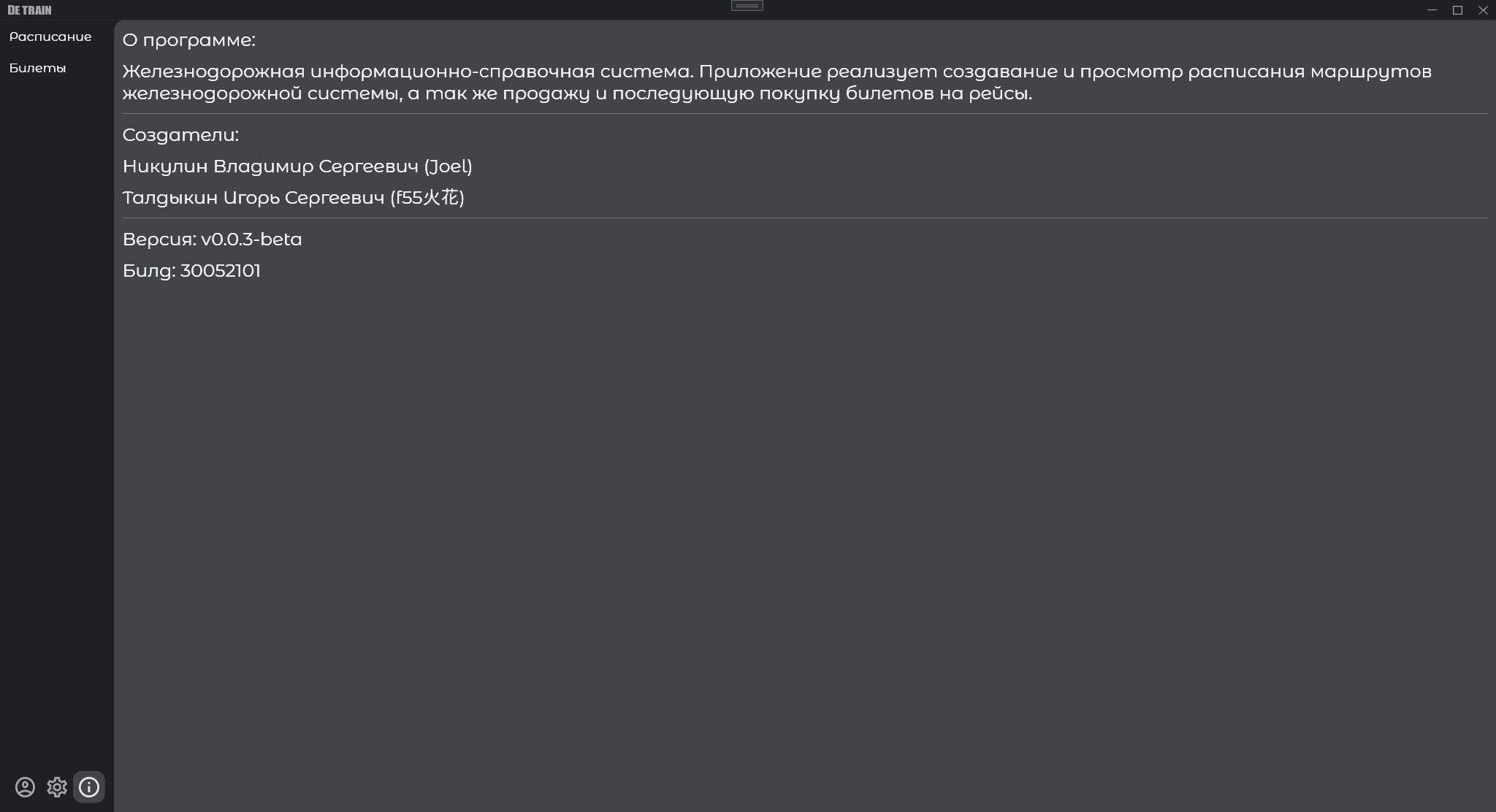


Рисунок 11 – Вкладка «Информация»

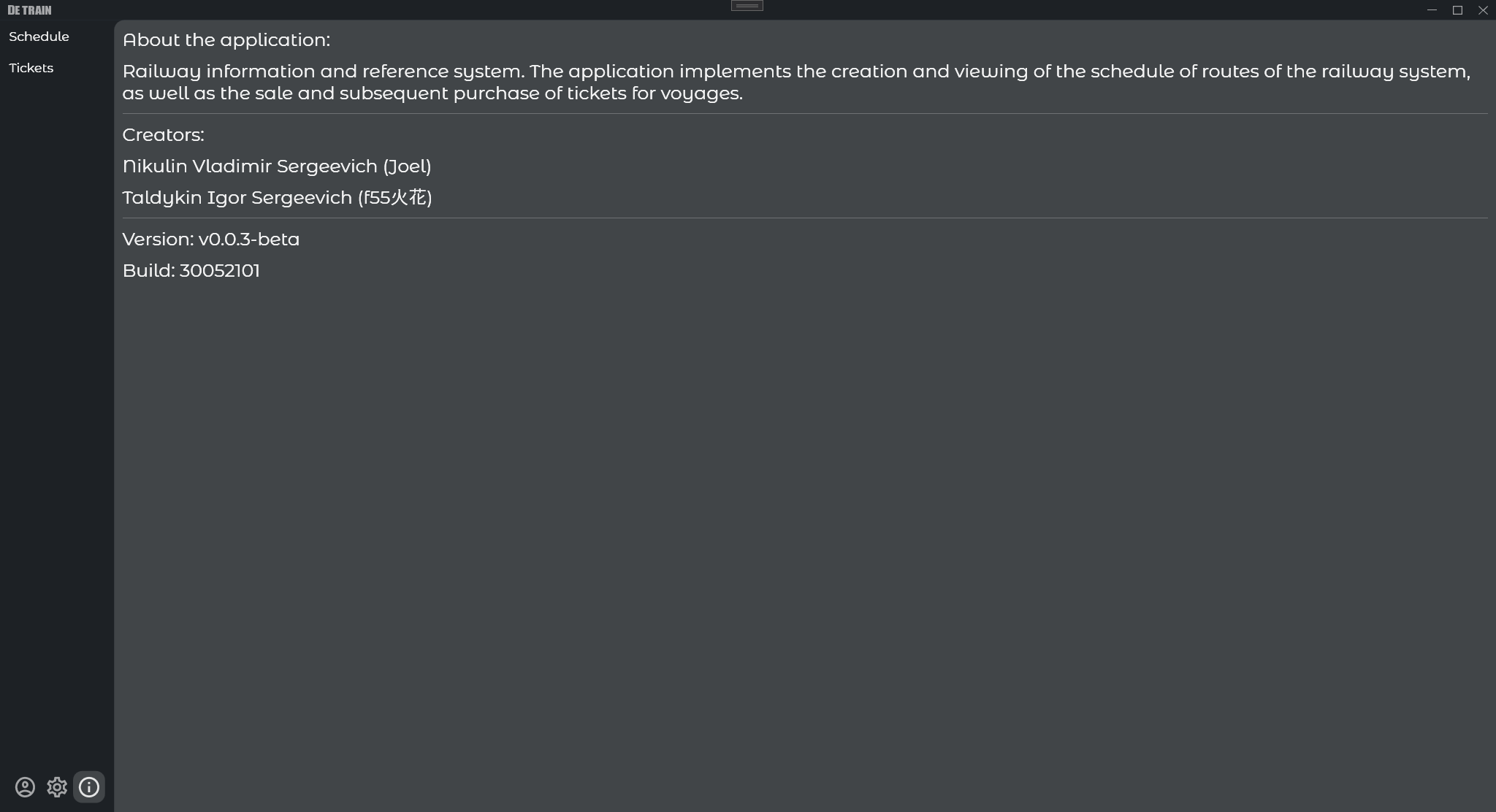


Рисунок 12 – Вкладка «Информация» на английском языке

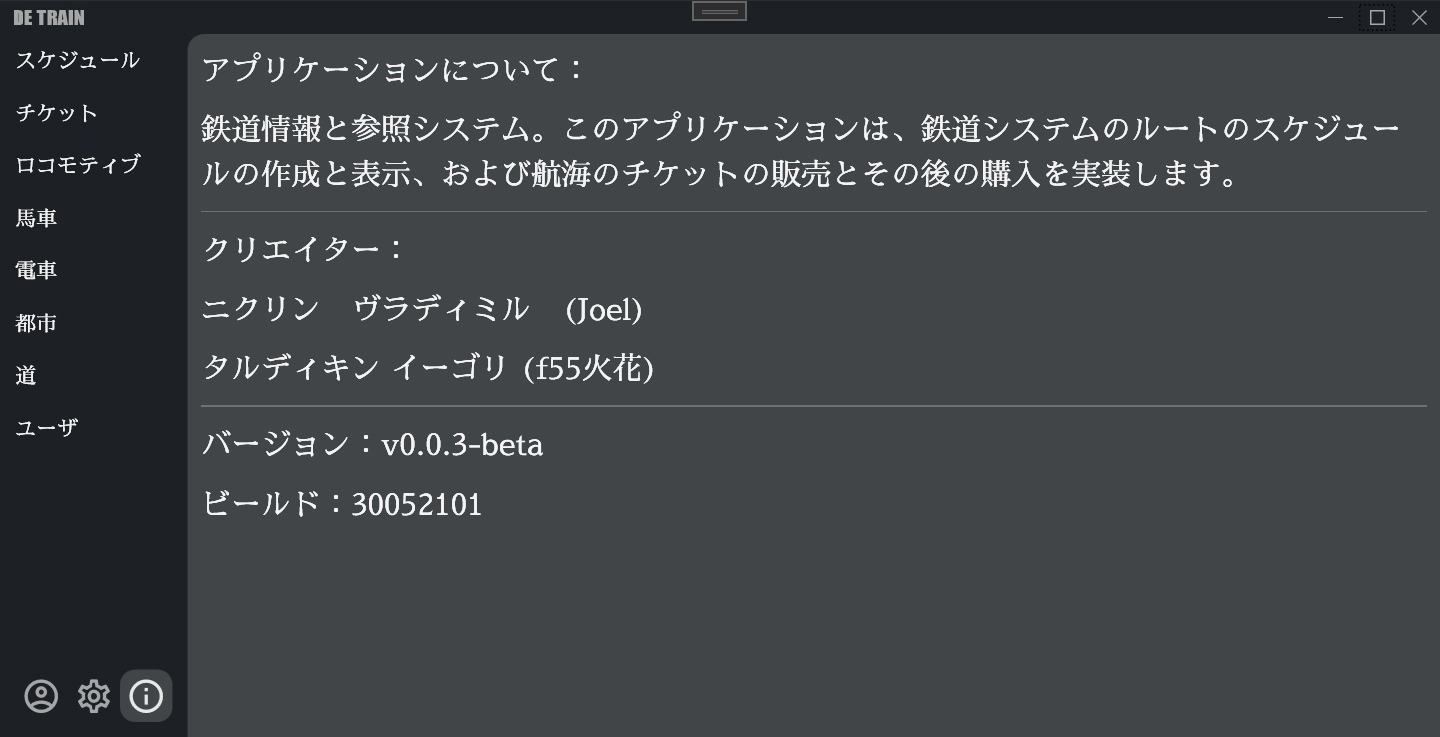


Рисунок 13 – Вкладка «Информация» на японском языке

### 2.2.5 Локализация приложения

Локализация проекта была разработана для 3 языков:

- русского;

- английского;

- японского.

Для каждого языка был создан словарь ресурсов, который находился в папке «Resources».

Отрывки кода из словарей ресурсов для всех языков представлены на листингах 2, 3 и 4.

Листинг 2 – Отрывок кода для русского языка

<!-- Labels -->

<sys:String x:Key="Text\_Language">Язык</sys:String>

<sys:String x:Key="Text\_PersonalCab">Личный кабинет</sys:String>

<sys:String x:Key="Text\_Rubles">рублей</sys:String>

<sys:String x:Key="Text\_Summ">Сумма</sys:String>

<sys:String x:Key="Text\_AboutLabel">О программе:</sys:String>

<sys:String x:Key="Text\_CreatorsLabel">Создатели:</sys:String>

<sys:String x:Key="Text\_VersionLabel">Версия: v0.0.3-beta</sys:String>

<sys:String x:Key="Text\_BuildLabel">Билд: 30052101</sys:String>

<sys:String x:Key="Text\_About">

Железнодорожная информационно-справочная система. Приложение реализует создавание и просмотр расписания маршрутов железнодорожной системы, а так же продажу и последующую покупку билетов на рейсы.

</sys:String>

<sys:String x:Key="Text\_Creators\_Nikulin">Никулин Владимир Сергеевич (Joel)</sys:String>

<sys:String x:Key="Text\_Creators\_Taldykin">Талдыкин Игорь Сергеевич (f55火花)</sys:String>

<!-- Messages -->

<sys:String x:Key="Text\_AddMoneyMessage">Деньги были успешно добавлены! На вашем счету:</sys:String>

<sys:String x:Key="Text\_NotEnoughMoney">Недостаточно денег для совершения покупки!</sys:String>

<sys:String x:Key="Text\_SuccessBuy">Вы успешно приобрели билет! Просмотр билетов осуществляется во вкладке "Билеты".</sys:String>

Листинг 3 – Отрывок кода для английского языка

<!-- Labels -->

<sys:String x:Key="Text\_Language">Language</sys:String>

<sys:String x:Key="Text\_PersonalCab">Profile</sys:String>

<sys:String x:Key="Text\_Rubles">RUB</sys:String>

<sys:String x:Key="Text\_Summ">Amount</sys:String>

<sys:String x:Key="Text\_AboutLabel">About the application:</sys:String>

<sys:String x:Key="Text\_CreatorsLabel">Creators:</sys:String>

<sys:String x:Key="Text\_VersionLabel">Version: v0.0.3-beta</sys:String>

<sys:String x:Key="Text\_BuildLabel">Build: 30052101</sys:String>

<sys:String x:Key="Text\_About">

Railway information and reference system. The application implements the creation and viewing of the schedule of routes of the railway system, as well as the sale and subsequent purchase of tickets for voyages.

</sys:String>

<sys:String x:Key="Text\_Creators\_Nikulin">Nikulin Vladimir Sergeevich (Joel)</sys:String>

<sys:String x:Key="Text\_Creators\_Taldykin">Taldykin Igor Sergeevich (f55火花)</sys:String>

<!-- Messages -->

<sys:String x:Key="Text\_AddMoneyMessage">Money has been successfully added! Your money:</sys:String>

<sys:String x:Key="Text\_NotEnoughMoney">Not enough money for a purchase!</sys:String>

<sys:String x:Key="Text\_SuccessBuy">You have successfully purchased your ticket! Tickets are viewed in the "Tickets" tab</sys:String>

Листинг 4 – Отрывок кода для японского языка

<!-- Labels -->

<sys:String x:Key="Text\_Language">言語</sys:String>

<sys:String x:Key="Text\_PersonalCab">パーソナルエリア</sys:String>

<sys:String x:Key="Text\_Rubles">ルーブル</sys:String>

<sys:String x:Key="Text\_Summ">金額</sys:String>

<sys:String x:Key="Text\_AboutLabel">アプリケーションについて：</sys:String>

<sys:String x:Key="Text\_CreatorsLabel">クリエイター：</sys:String>

<sys:String x:Key="Text\_VersionLabel">バージョン：v0.0.3-beta</sys:String>

<sys:String x:Key="Text\_BuildLabel">ビールド：30052101</sys:String>

<sys:String x:Key="Text\_About">

鉄道情報と参照システム。このアプリケーションは、鉄道システムのルートのスケジュールの作成と表示、および航海のチケットの販売とその後の購入を実装します。

</sys:String>

<sys:String x:Key="Text\_Creators\_Nikulin">ニクリン　ヴラディミル　(Joel)</sys:String>

<sys:String x:Key="Text\_Creators\_Taldykin">タルディキン イーゴリ (f55火花)</sys:String>

<!-- Messages -->

<sys:String x:Key="Text\_AddMoneyMessage">お金が正常に追加されました！あなたのお金：</sys:String>

<sys:String x:Key="Text\_NotEnoughMoney">購入するのに十分なお金がありません！</sys:String>

<sys:String x:Key="Text\_SuccessBuy">チケットの購入に成功しました。 チケットは「チケット」タブで見ることができます。</sys:String>

Вся логика переключения языка была прописана в файле App.cs, а именно в параметре Language имеющий методы «get;» и «set;». Код данного параметра представлен в листинге 5.

Листинг 5.

public static CultureInfo Language

{get{return System.Threading.Thread.CurrentThread.CurrentUICulture; }

set

{ if (value == null) throw new ArgumentNullException("value");

if (value == System.Threading.Thread.CurrentThread.CurrentUICulture) return;

System.Threading.Thread.CurrentThread.CurrentUICulture = value;

ResourceDictionary dict = new ResourceDictionary();

switch (value.Name)

{ case "en-US":

case "ja-JP":

dict.Source = new Uri(String.Format("Resources/Languages/Lang.{0}.xaml", value.Name), UriKind.Relative);

break;

default:

dict.Source = new Uri("Resources/Languages/Lang.xaml", UriKind.Relative);

break; }

ResourceDictionary ResourceDictionary oldDict = (from d in Application.Current.Resources.MergedDictionaries where d.Source != null && d.Source.OriginalString.StartsWith("Resources/Languages/Lang.") select d).First();

if (oldDict != null)

{ int ind = Application.Current.Resources.MergedDictionaries.IndexOf(oldDict);

Application.Current.Resources.MergedDictionaries.Remove(oldDict);

Application.Current.Resources.MergedDictionaries.Insert(ind, dict);

}else{Application.Current.Resources.MergedDictionaries.Add(dict); }

LanguageChanged(Application.Current, new EventArgs());}

}

Примеры отображения английского языка показаны на рисунках 14 и 15.

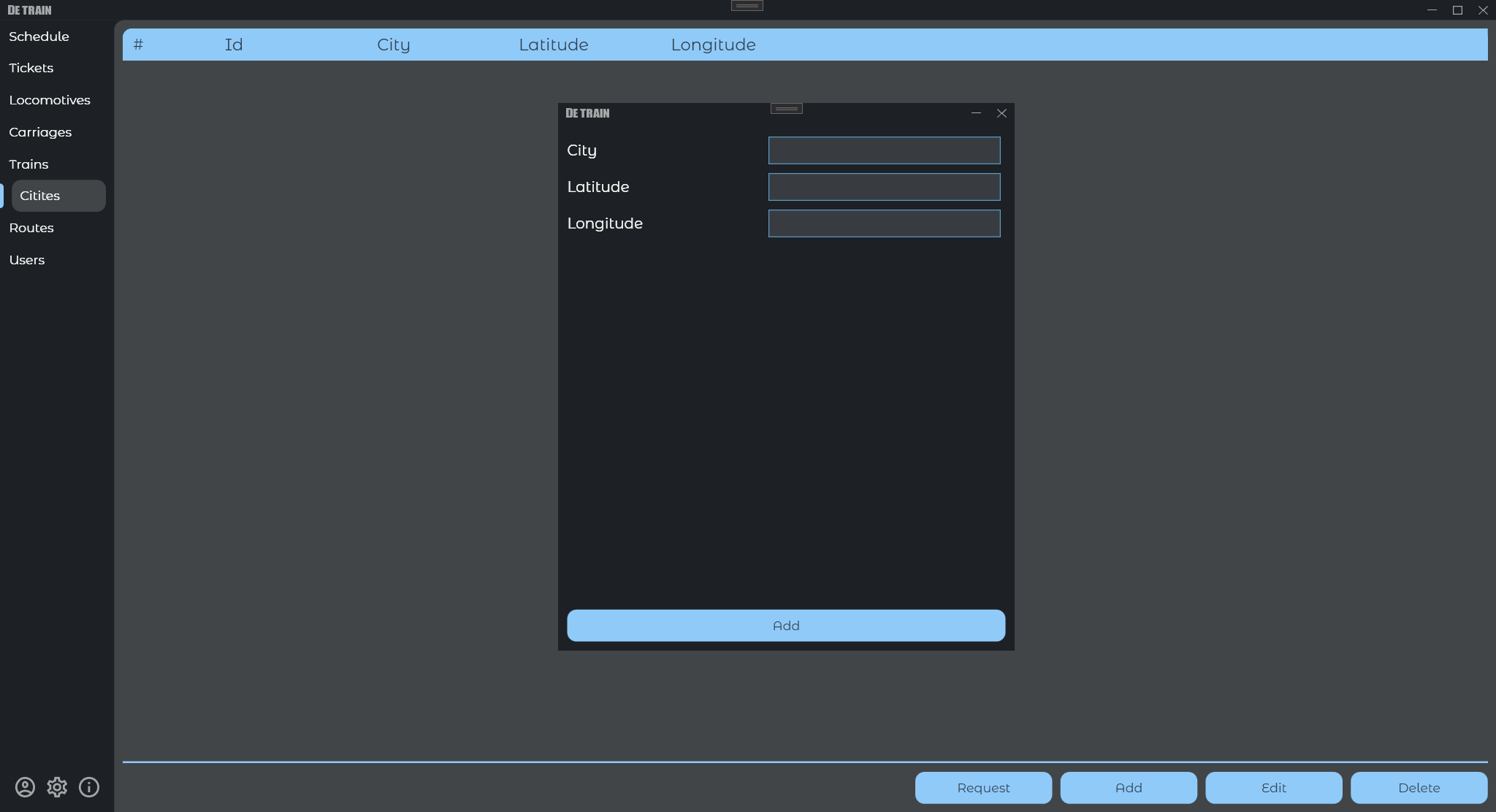


Рисунок 14 – Добавление городов на английском языке

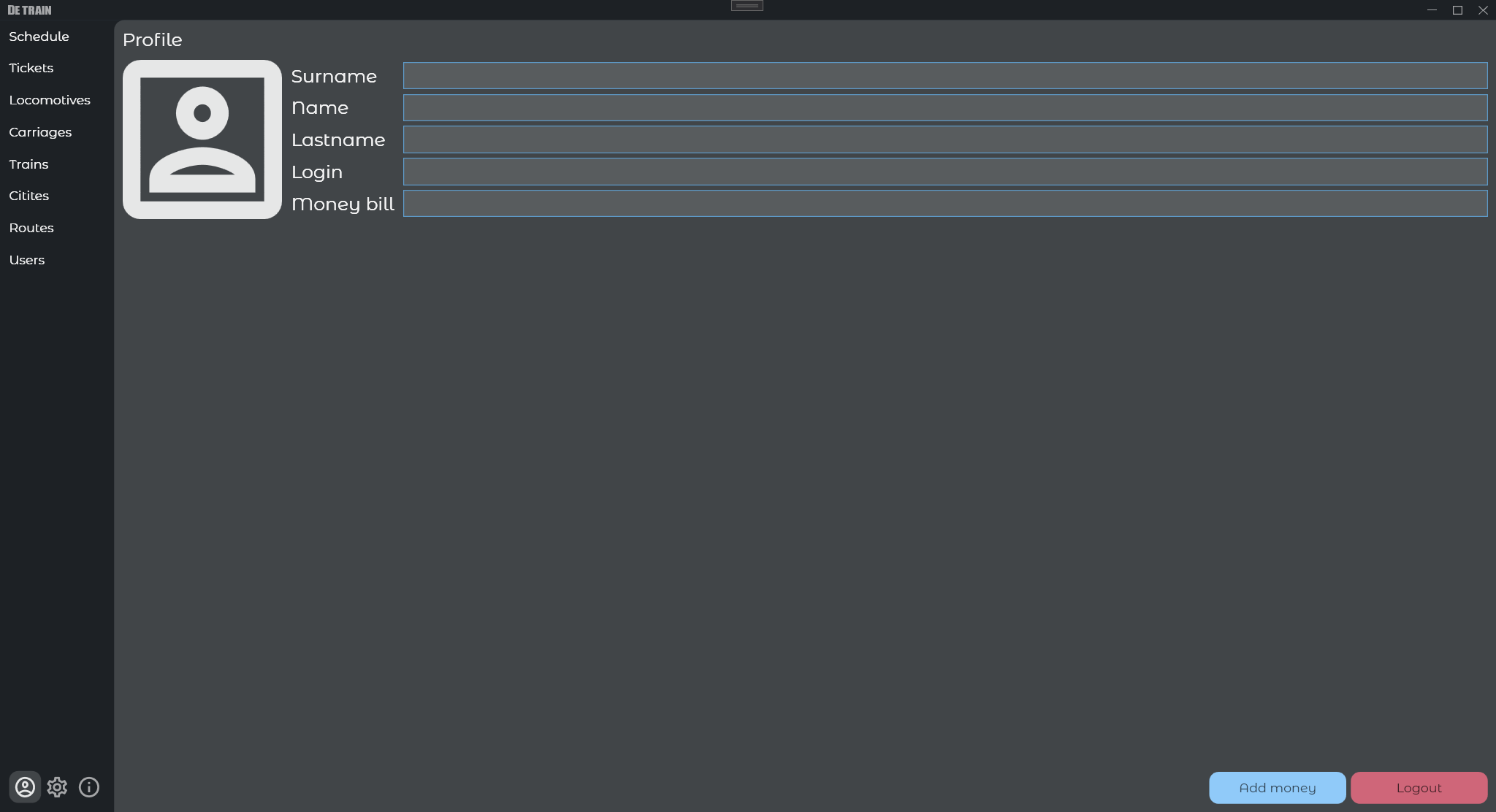


Рисунок 15 – Личный кабинет на английском языке

Примеры отображения японского языка показаны на рисунках 16 и 17.

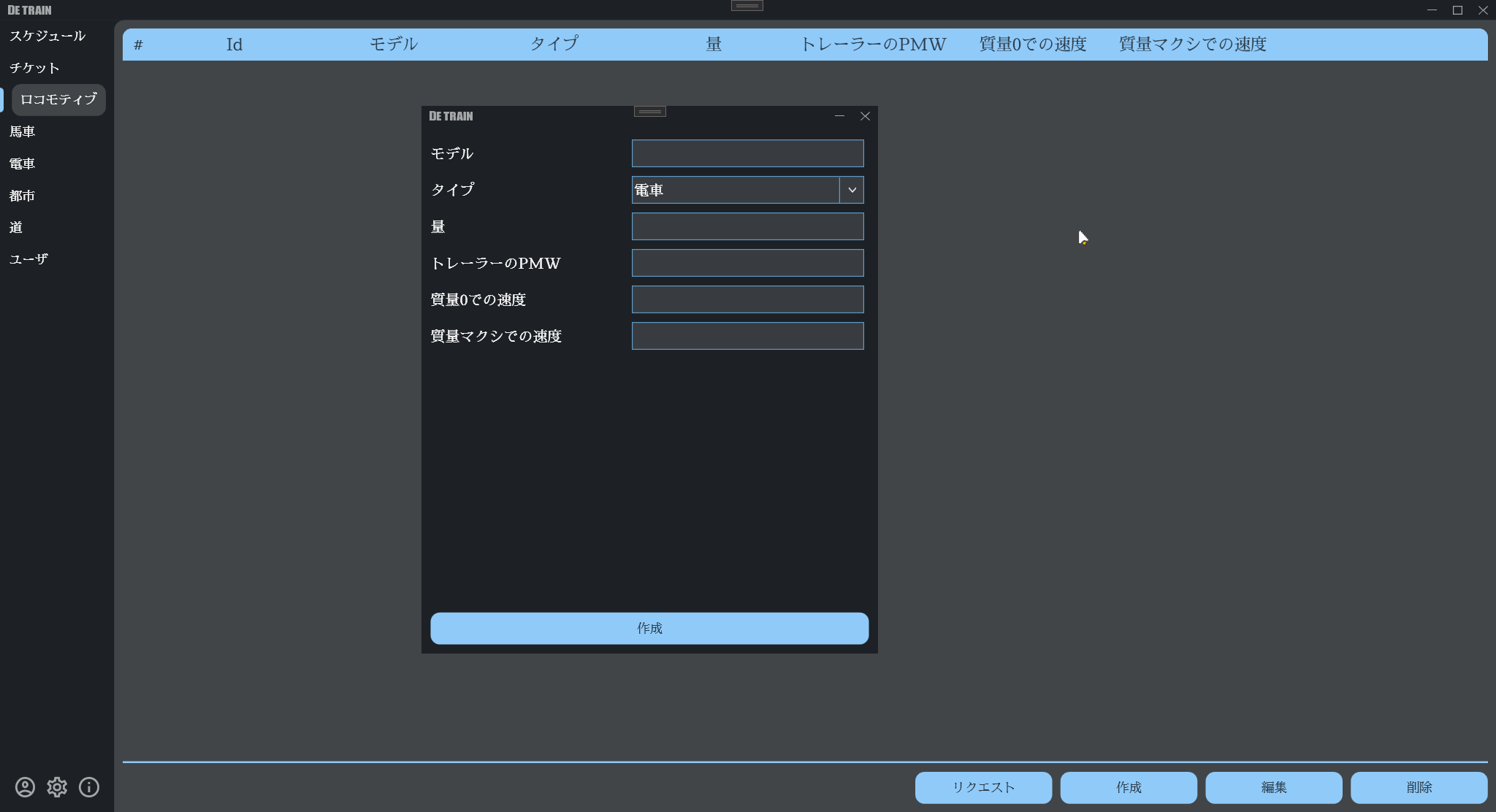


Рисунок 16 – Добавление локомотивов на японском языке

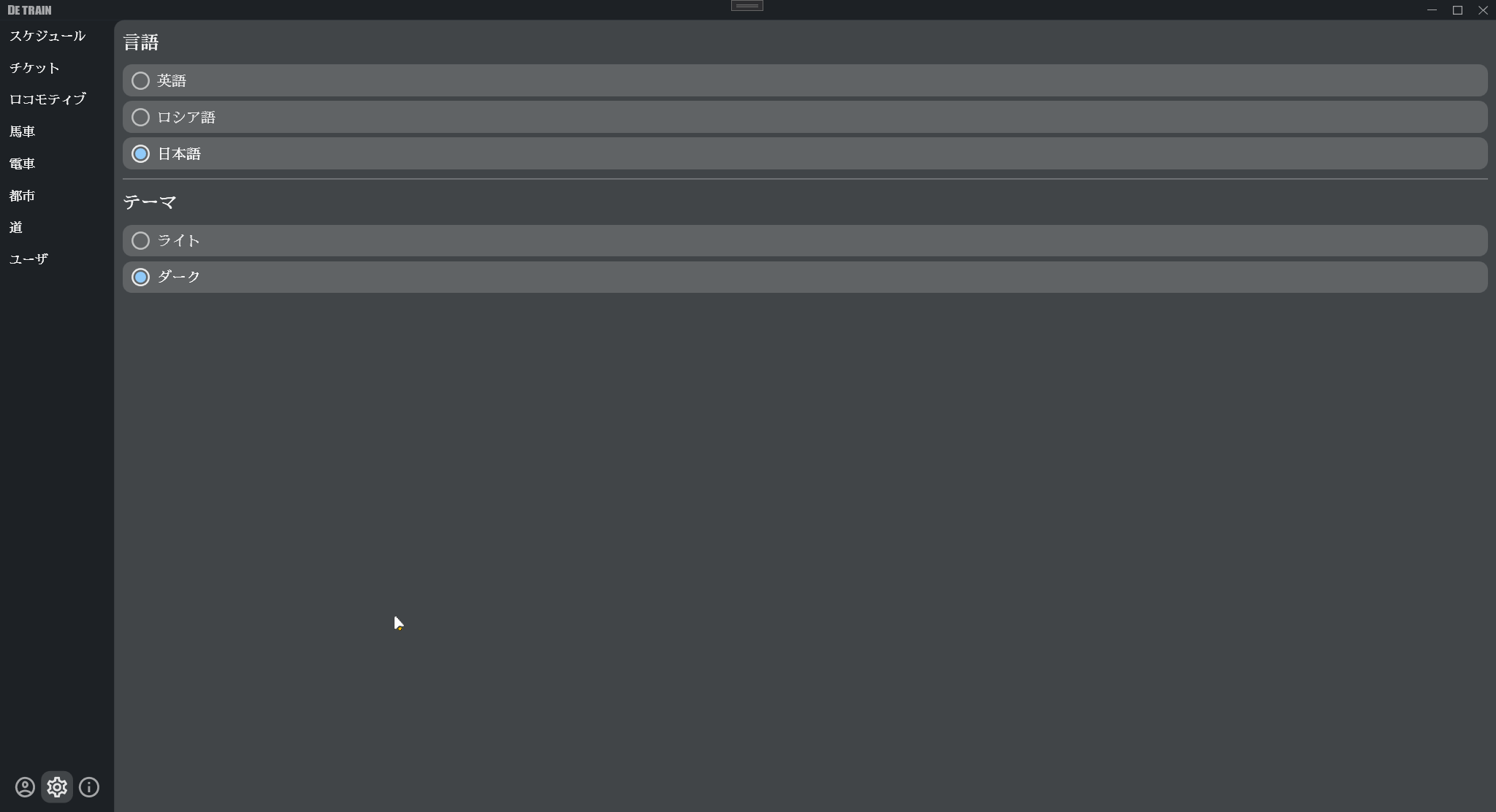


Рисунок 17 – Окно настроек на японском языке

# 3 Тестирование железнодорожной информационно-справочной системы

Тестирование проводилось при всех моделях использования – администратор и покупатель. Для начала было протестировано окно регистрации. Как можно увидеть из рисунка 18, регистрация проходит успешно при введении соответствующих данных в соответствующем им формате, иначе выводится ошибка.

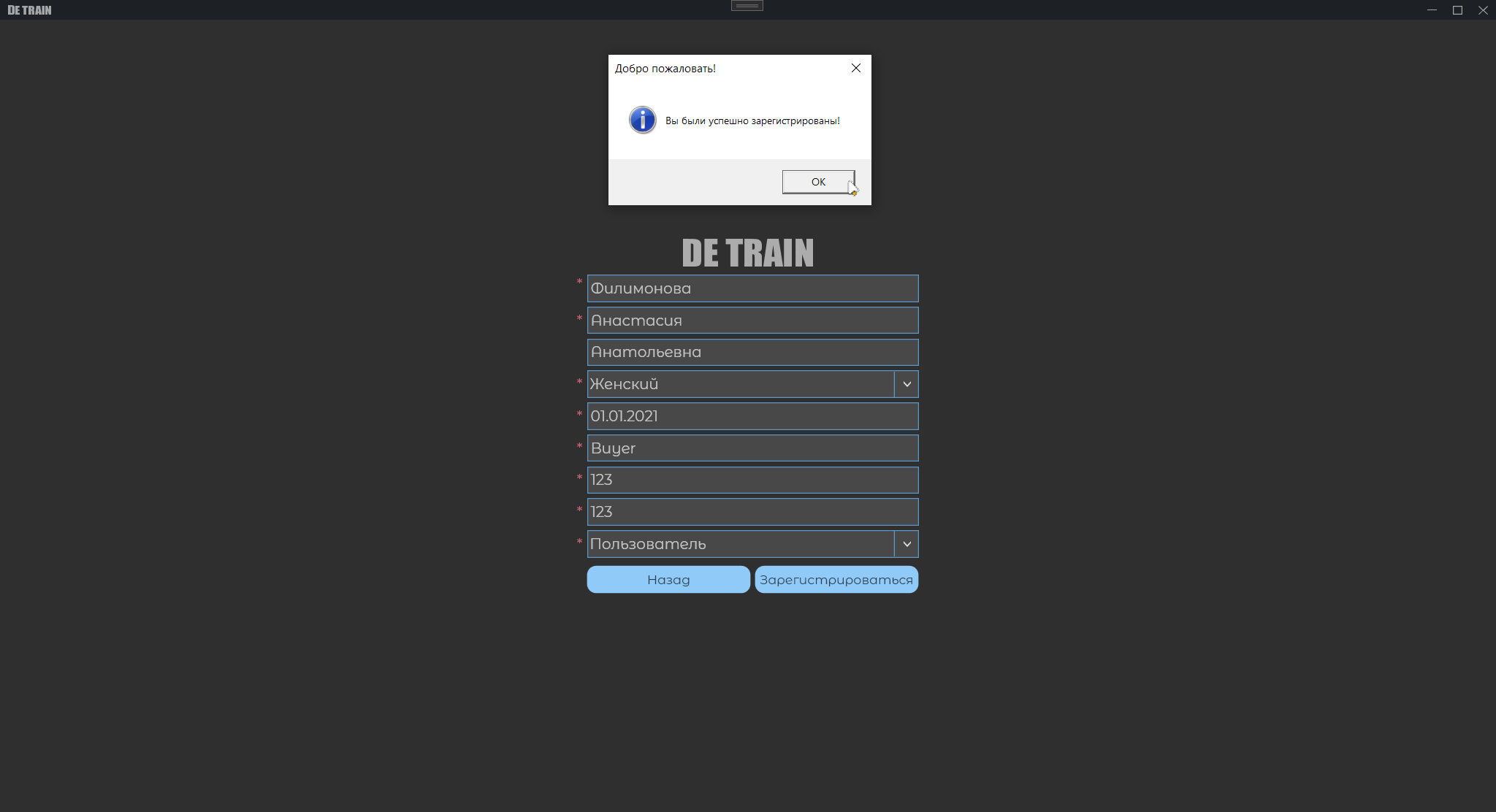


Рисунок 18 – Поле регистрации

Далее приложение протестировано на ввод неправильных данных при входе в аккаунт. Как видно из рисунка 19, логин был введен неверно, в следствии чего приложение вывело ошибку.

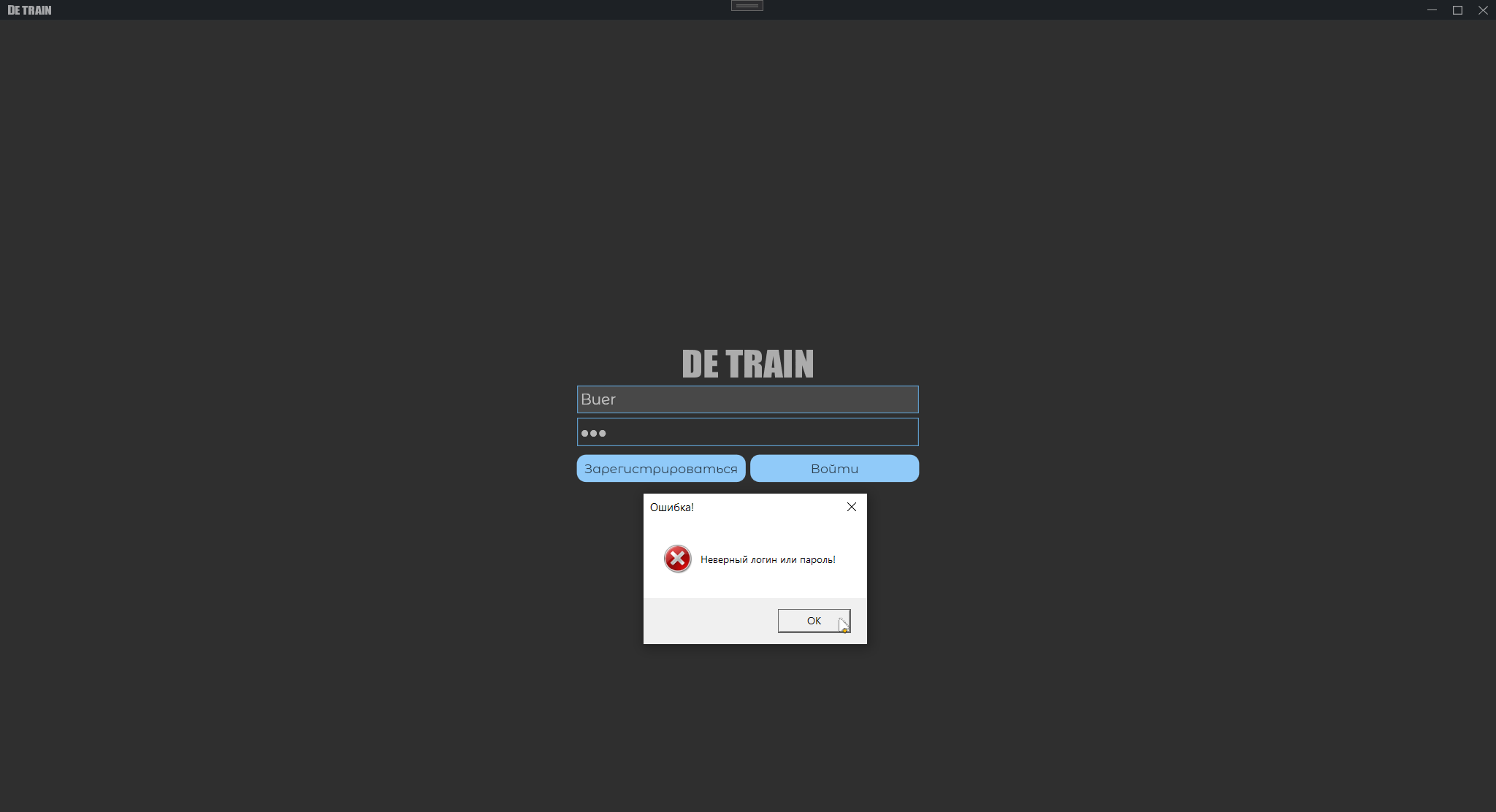


Рисунок 19 – Проверка ввода данных

Далее в систему зашли под администратором и протестировали создание записи в две из таблиц – таблицу с городами и расписанием. Выбор таблиц для показного тестирования обусловлен тем, что без добавления записей в промежуточные таблицы между городами и расписанием, запись в таблицу с расписанием добавить невозможно. Как видно из рисунков 20 и 21, добавление записей в таблицу с городами и расписанием работает.

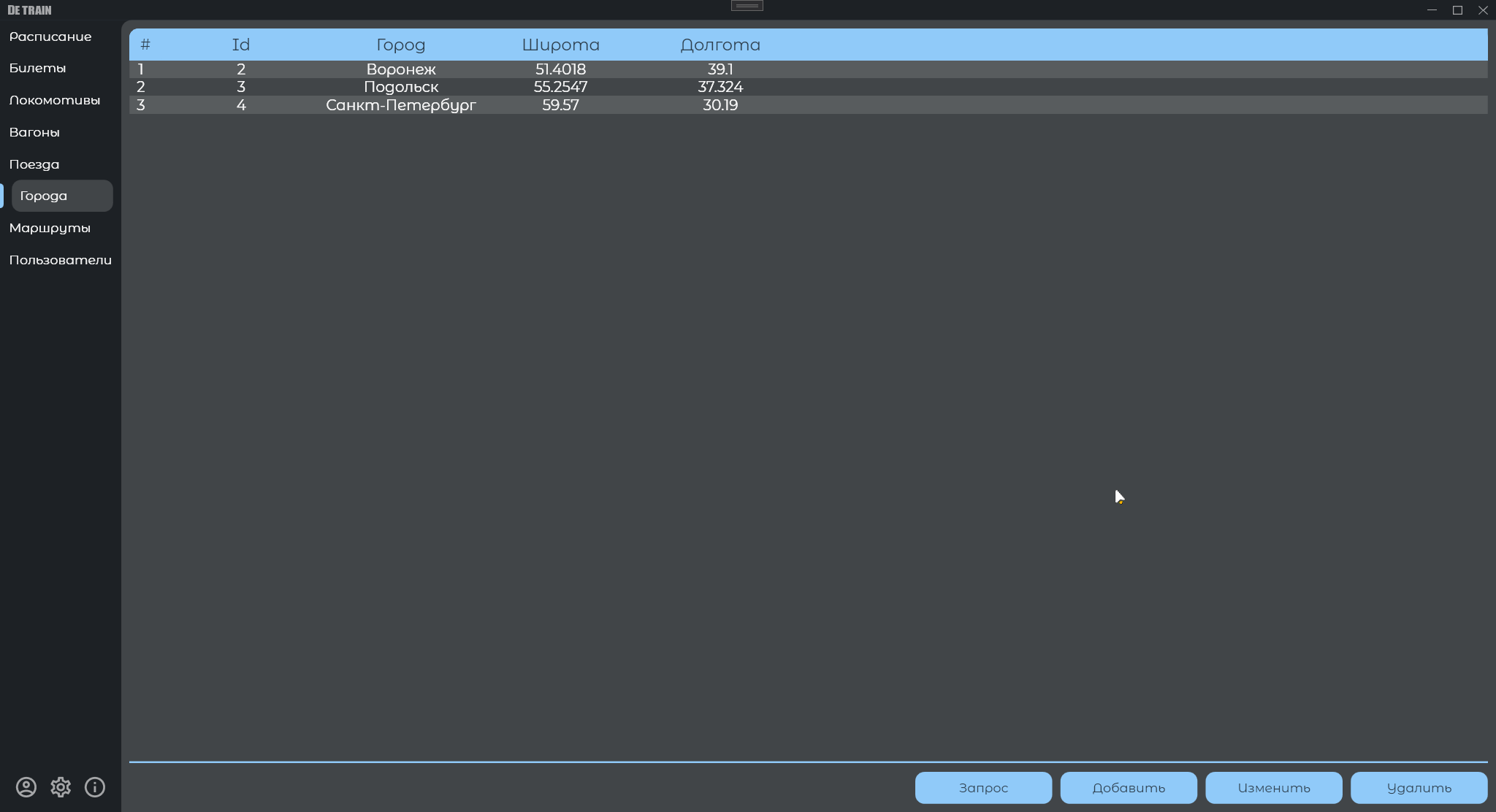


Рисунок 20 – Заполнение таблицы городов

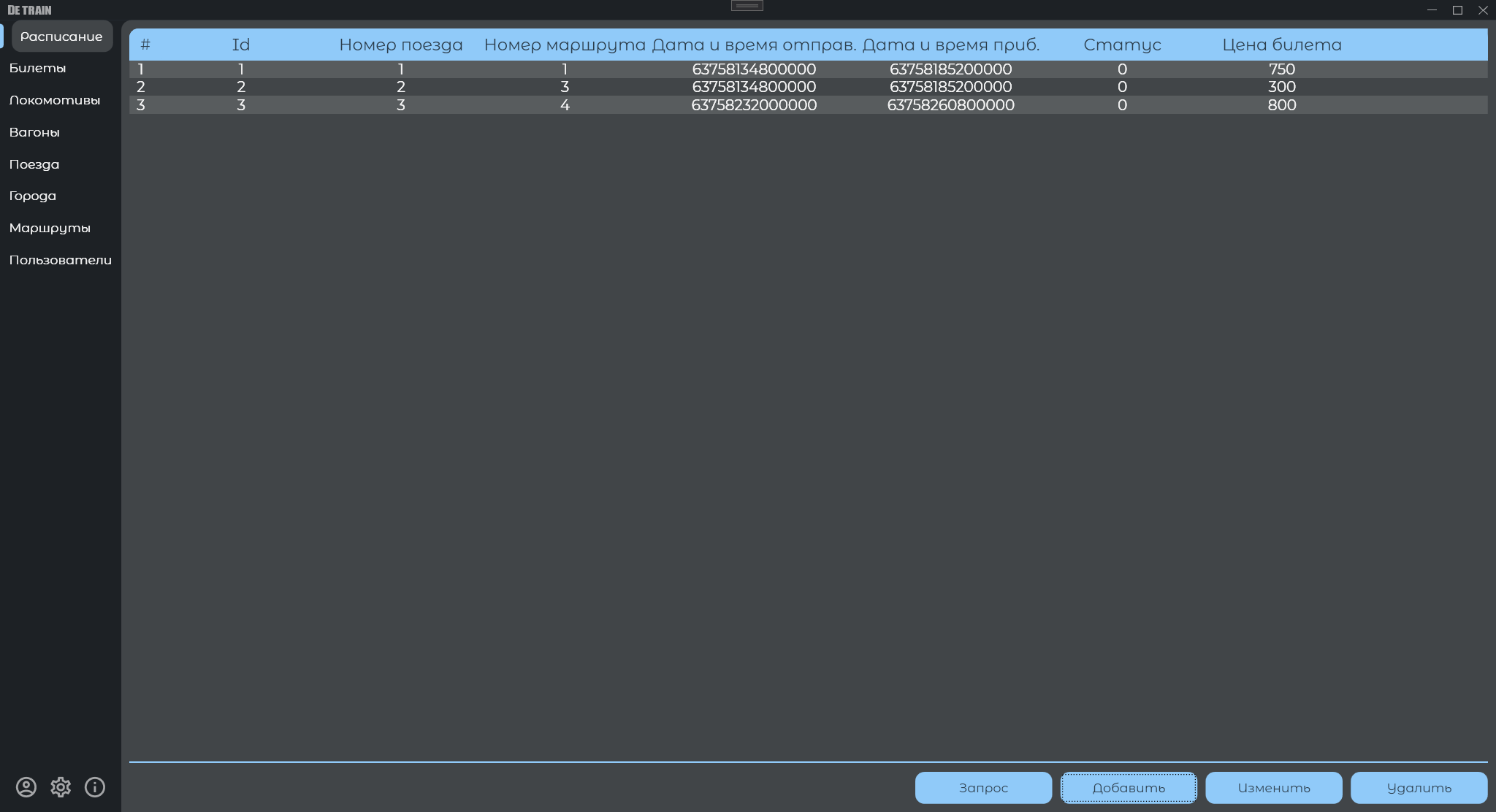


Рисунок 21 – Заполнение таблицы расписания

Далее было протестировано удаление и изменение в одной из таблиц – таблице маршрутов, при выборе элемента таблицы. Как видно из рисунков 22 и 23, удаление работает. И так же на рисунках 24 и 25 работает изменение.

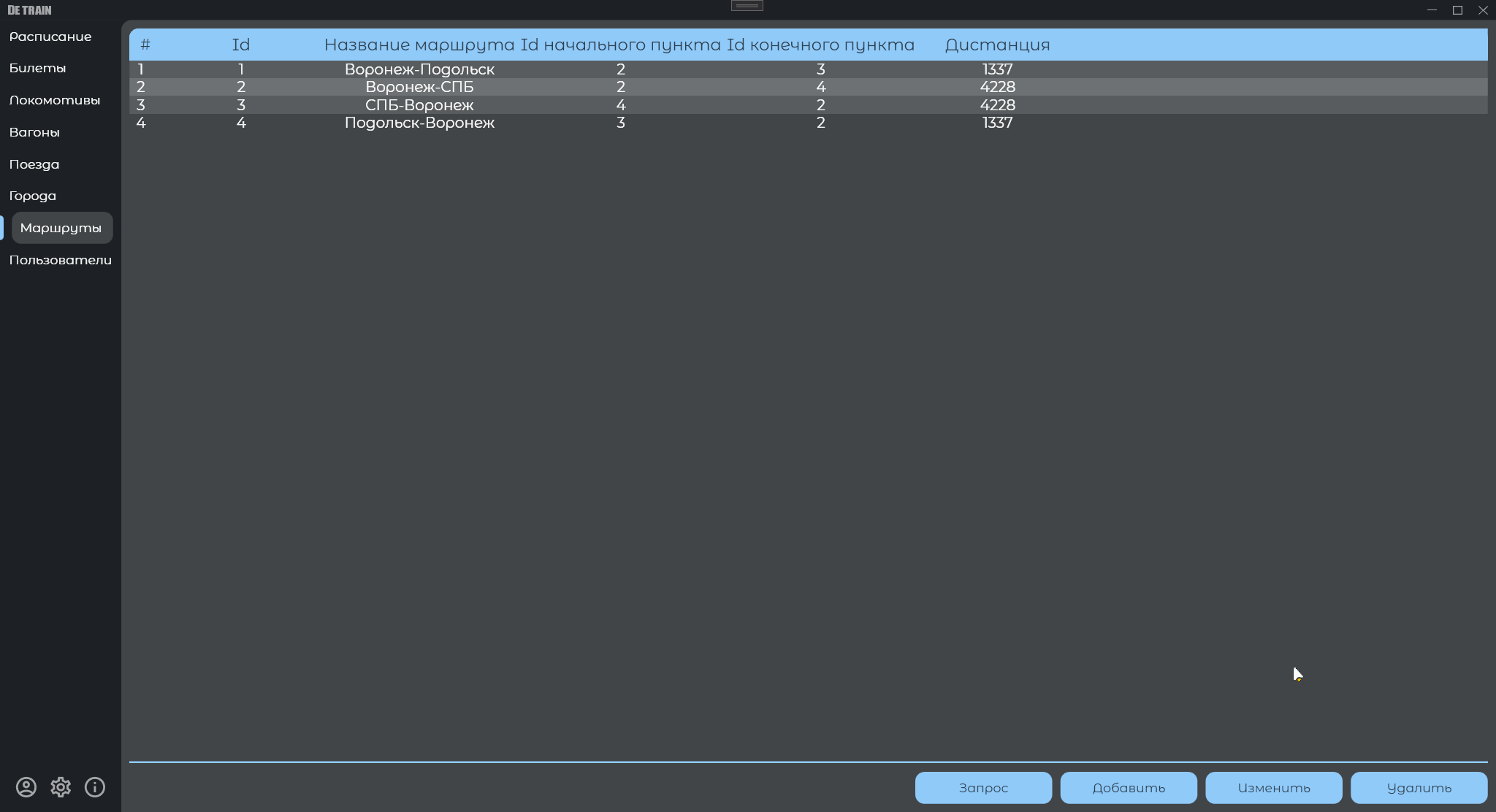


Рисунок 22 – Элемент таблицы выбран

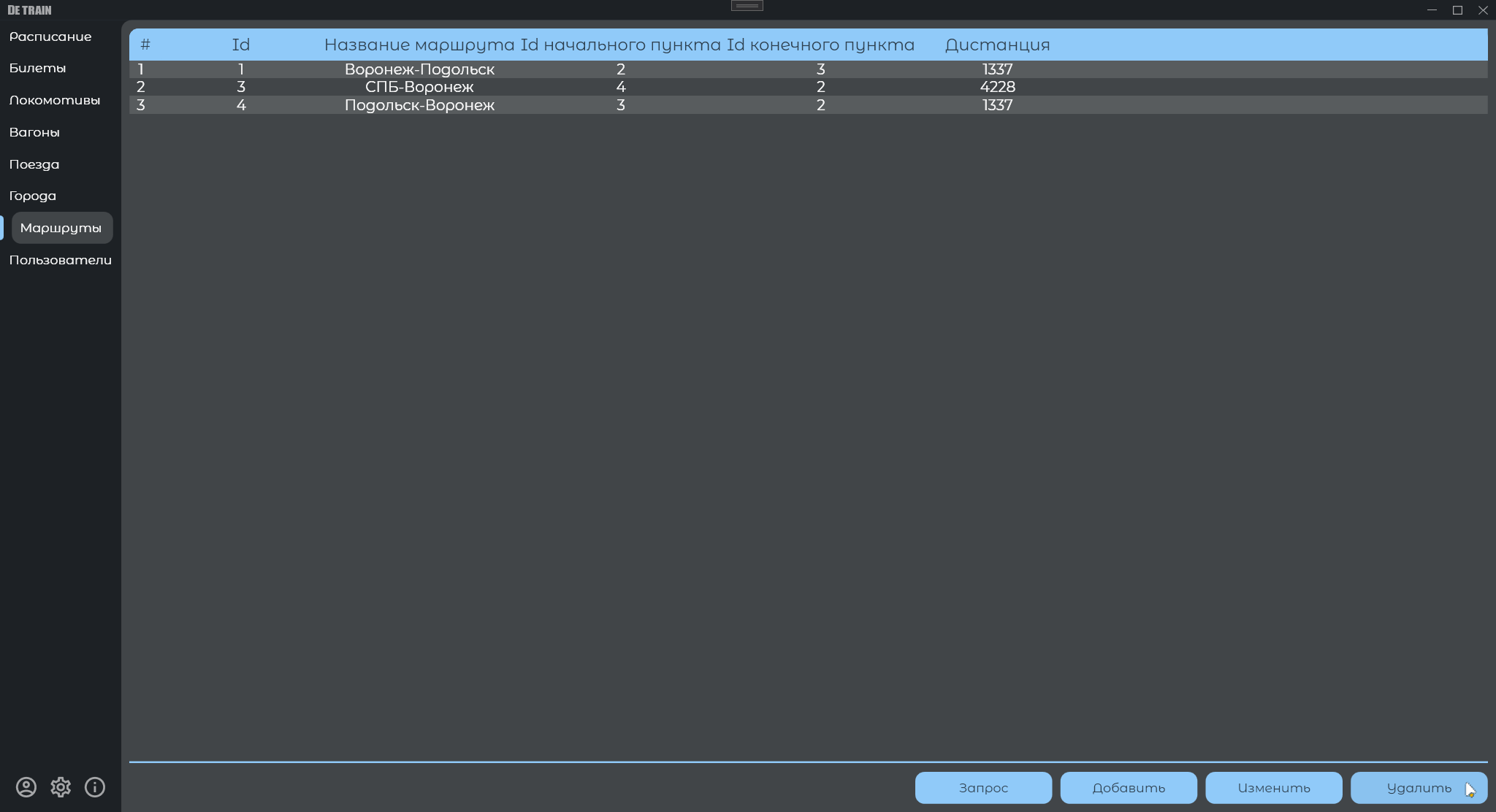


Рисунок 23 – Выбранный элемент удален

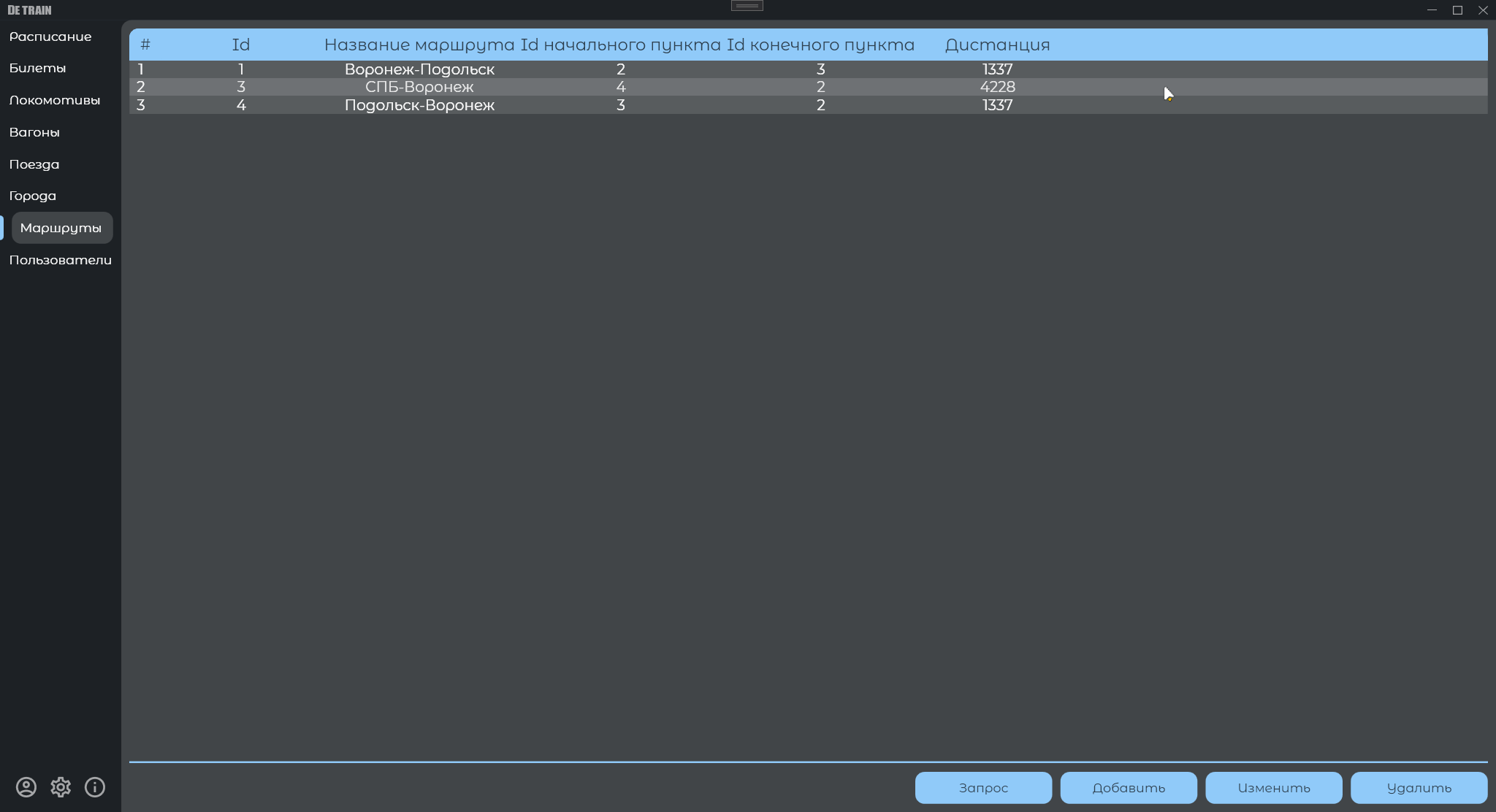


Рисунок 24 – Элемент таблицы выбран

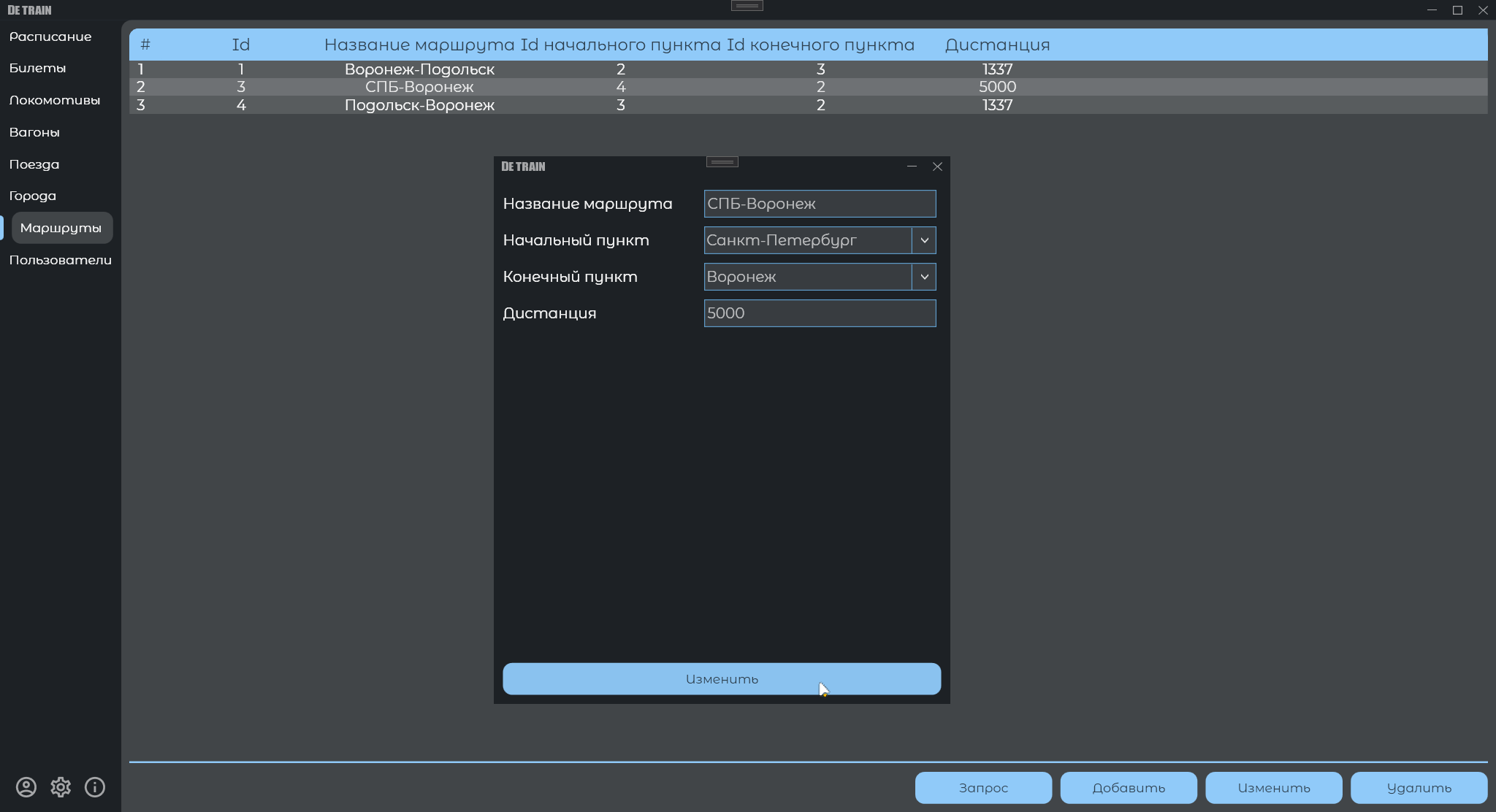


Рисунок 25 – Выбранный элемент изменён

Далее в систему зашли под пользователем и протестировали покупку билетов в таблице расписаний. Сначала попробовали купить билет на который хватает денег (на счету 750 рублей), а после купить билет за 800 рублей. Как видно из рисунков 26, 27 и 28, покупка проходит успешно и не успешно в соответствующих случаях.

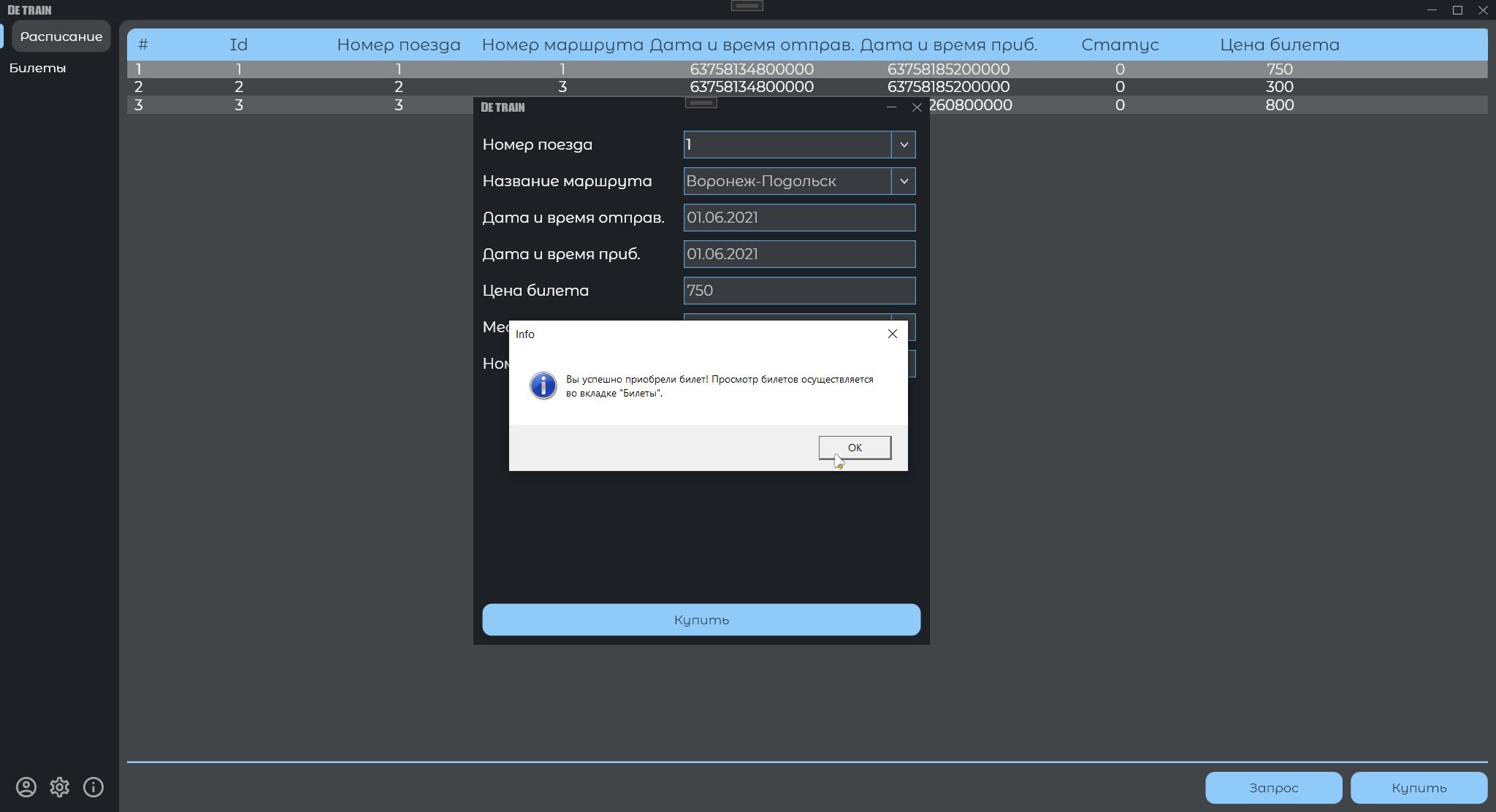


Рисунок 26 – Билет куплен

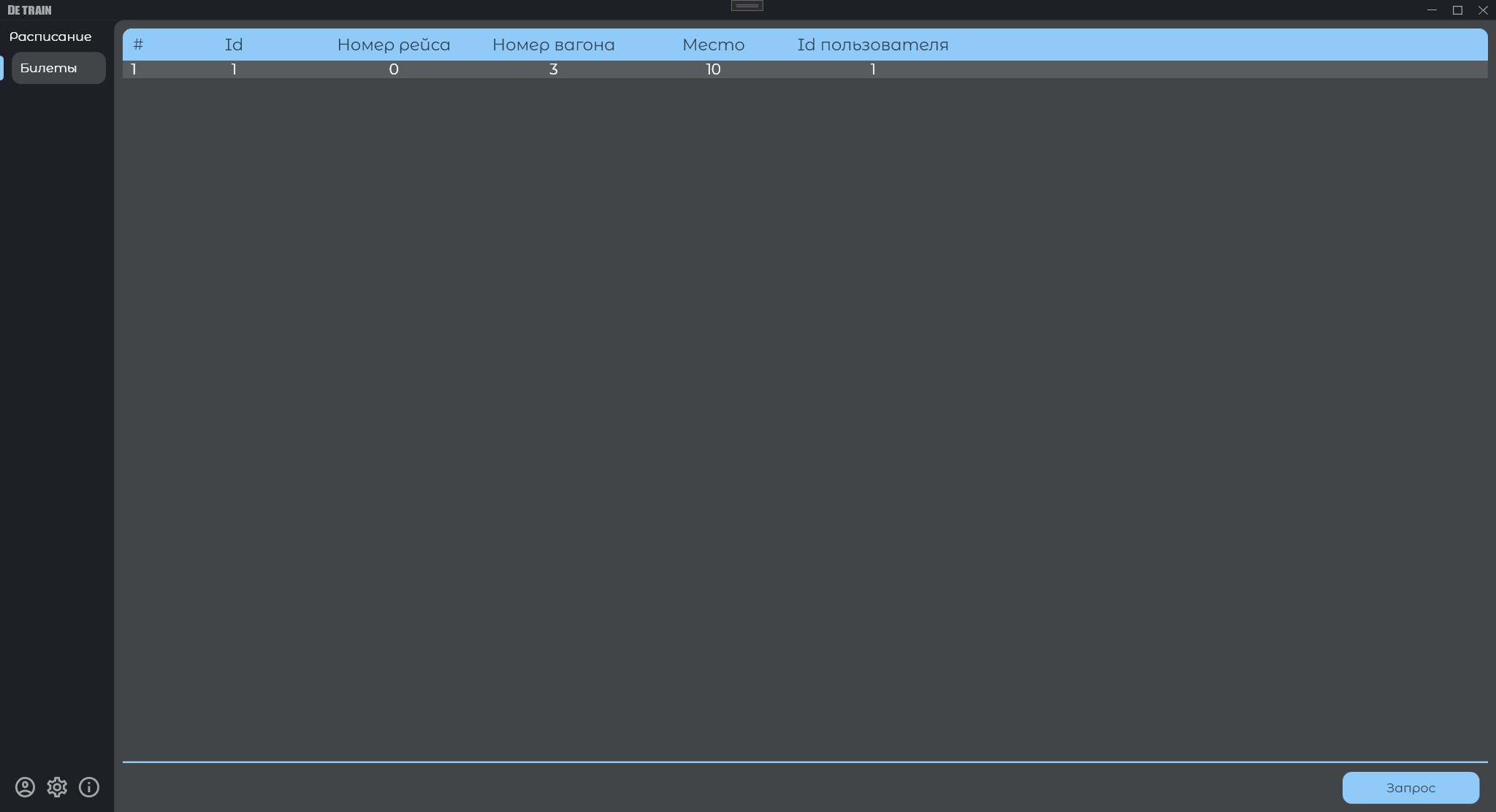


Рисунок 27 – Билет добавлен во вкладку

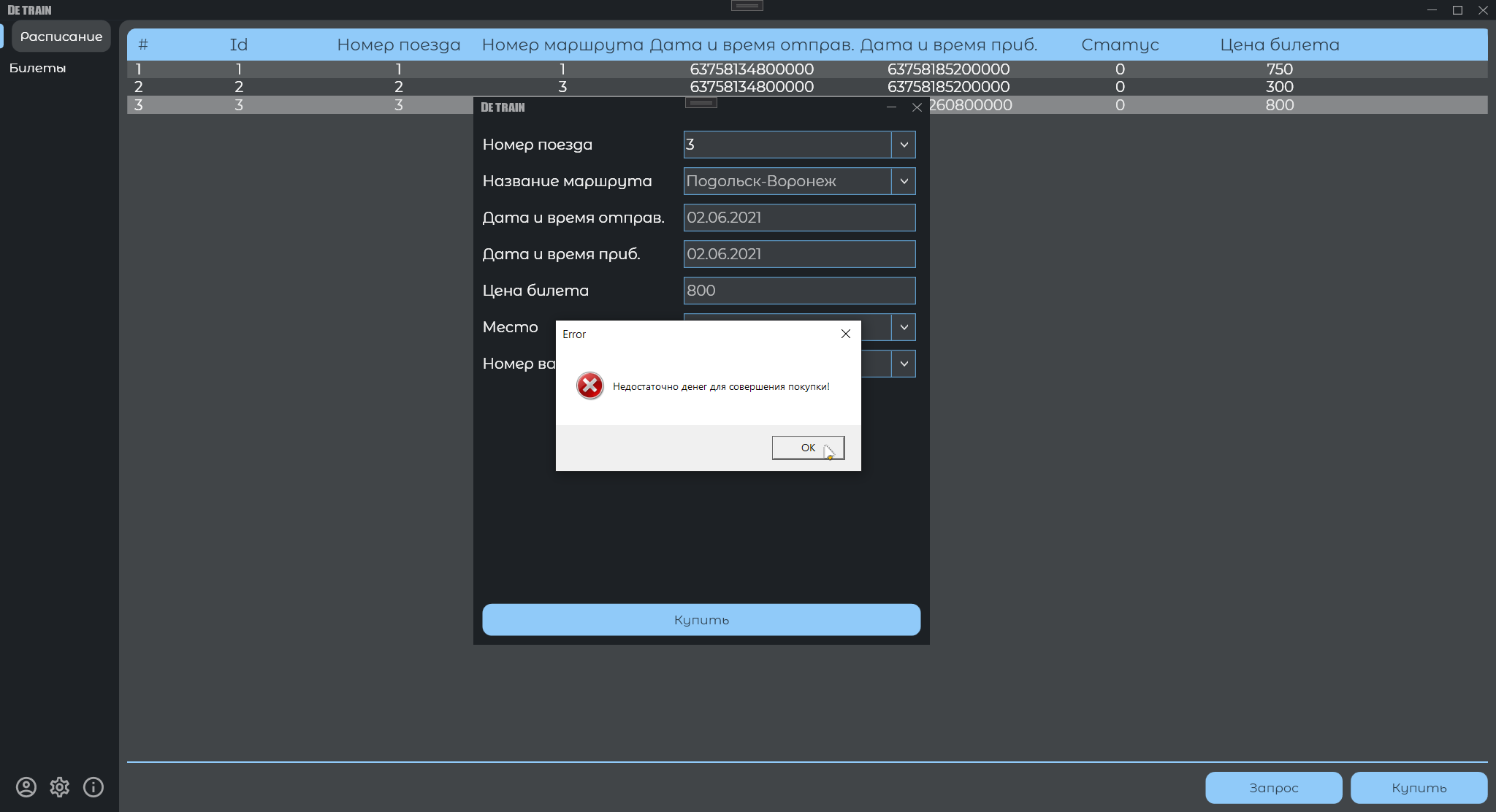


Рисунок 28 – Ошибка о недостатке средств

Далее был протестирован запрос для расписания, поиск по названию маршрута. Как видно из рисунка 29, запрос работает и в таблице остались лишь подходящие по запросу записи.

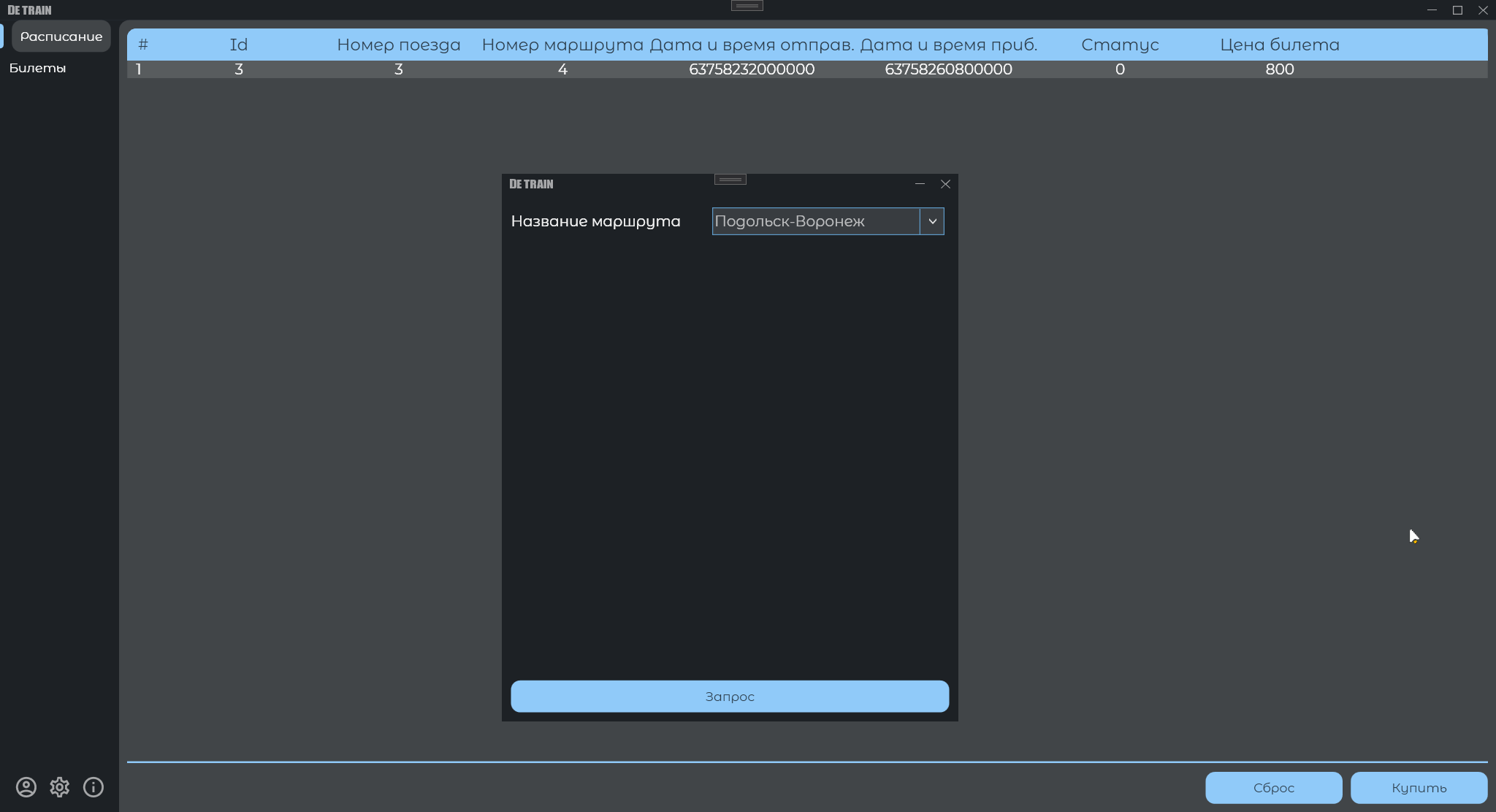


Рисунок 29 – Запрос сработал

Далее было протестирован дополнительный функционал приложения, были переключены языки, а так же светлая тема приложения. Как видно на рисунках 30 и 31, язык приложения переключается на соответствующий. А так же на рисунке 32 представлена светлая тема приложения.

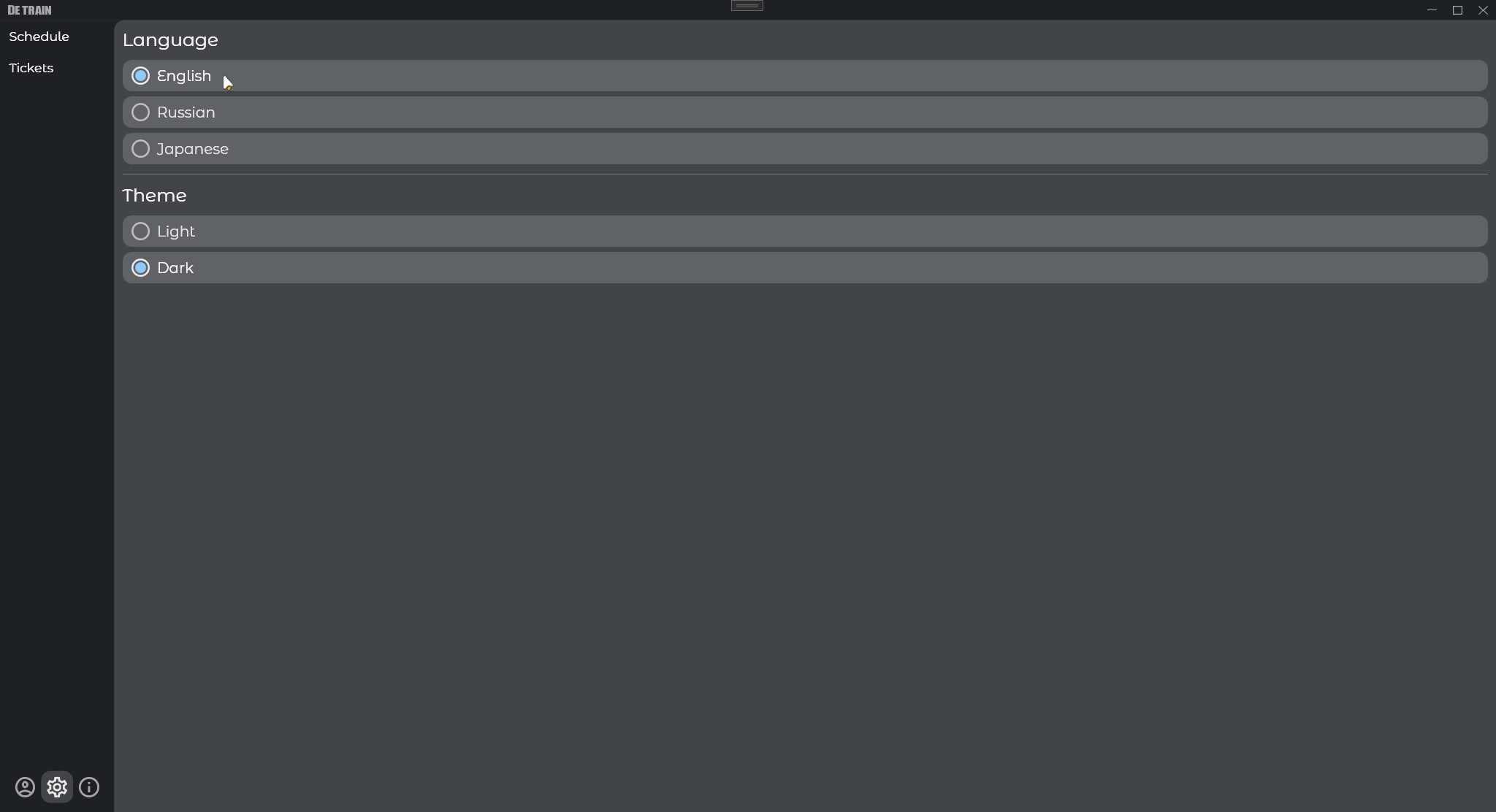


Рисунок 30 – Переключение языка на английский

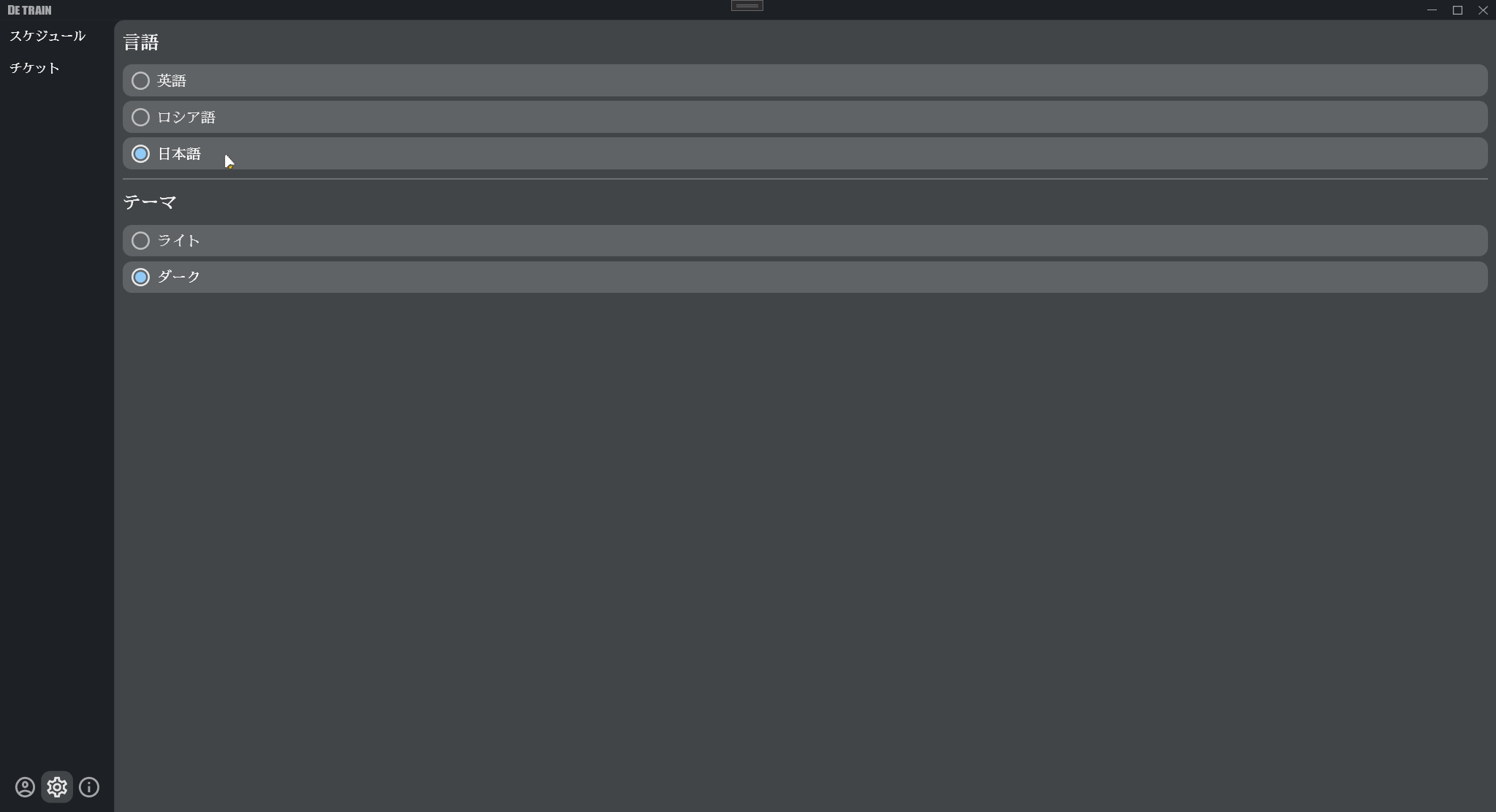


Рисунок 31 – Переключение языка на японский

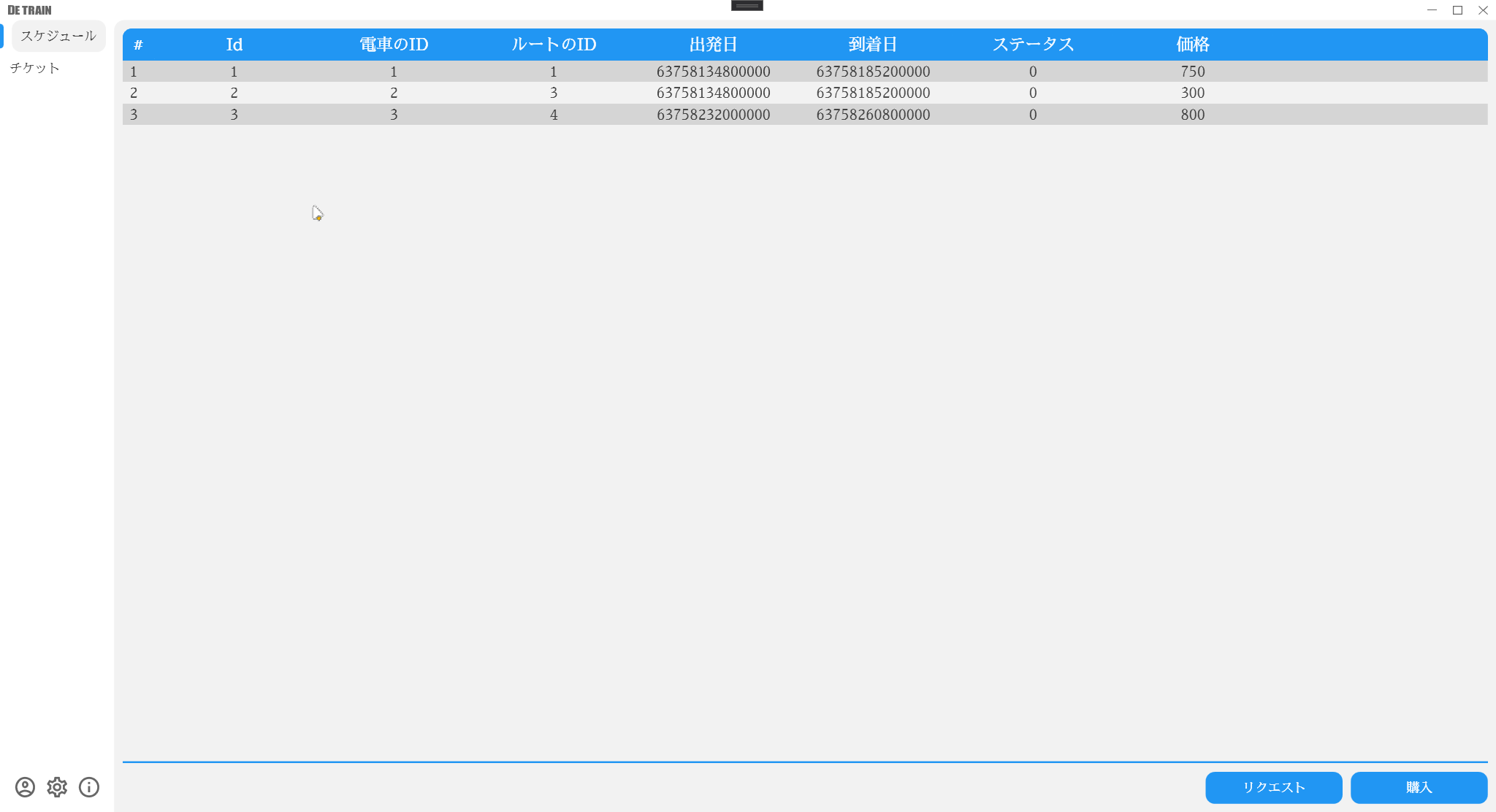


Рисунок 32 – Светлая тема приложения

Заключительным этапом было тестирование пополнения счёта аккаунта. На аккаунте сначала был нулевой баланс, при нажатии «Пополнить счёт» появляется диалоговое окно в котором указали нужную сумму для пополнения. Как видно из рисунков 33 и 34 деньги были успешно добавлены на баланс.

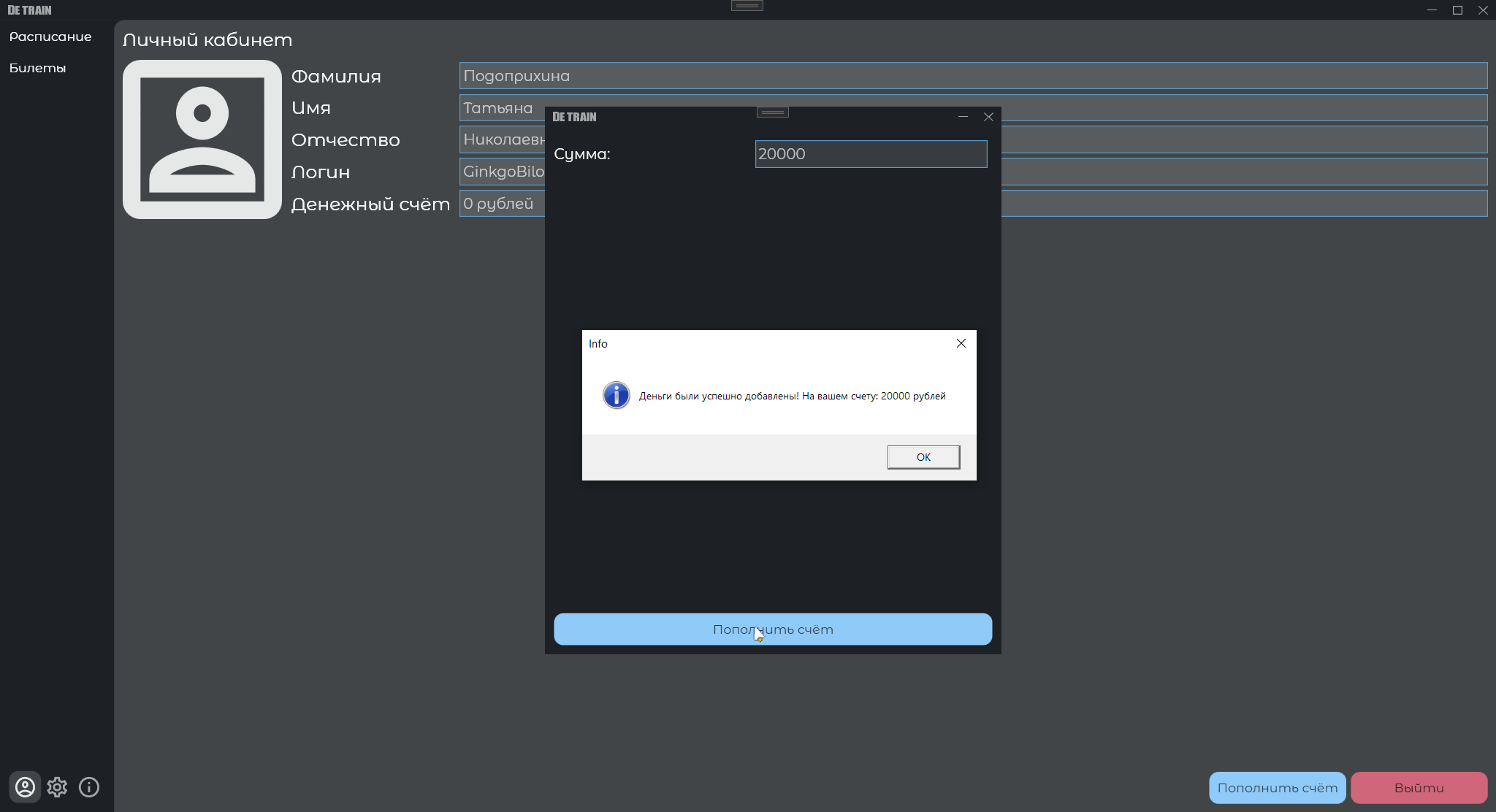


Рисунок 33 – Деньги добавлены

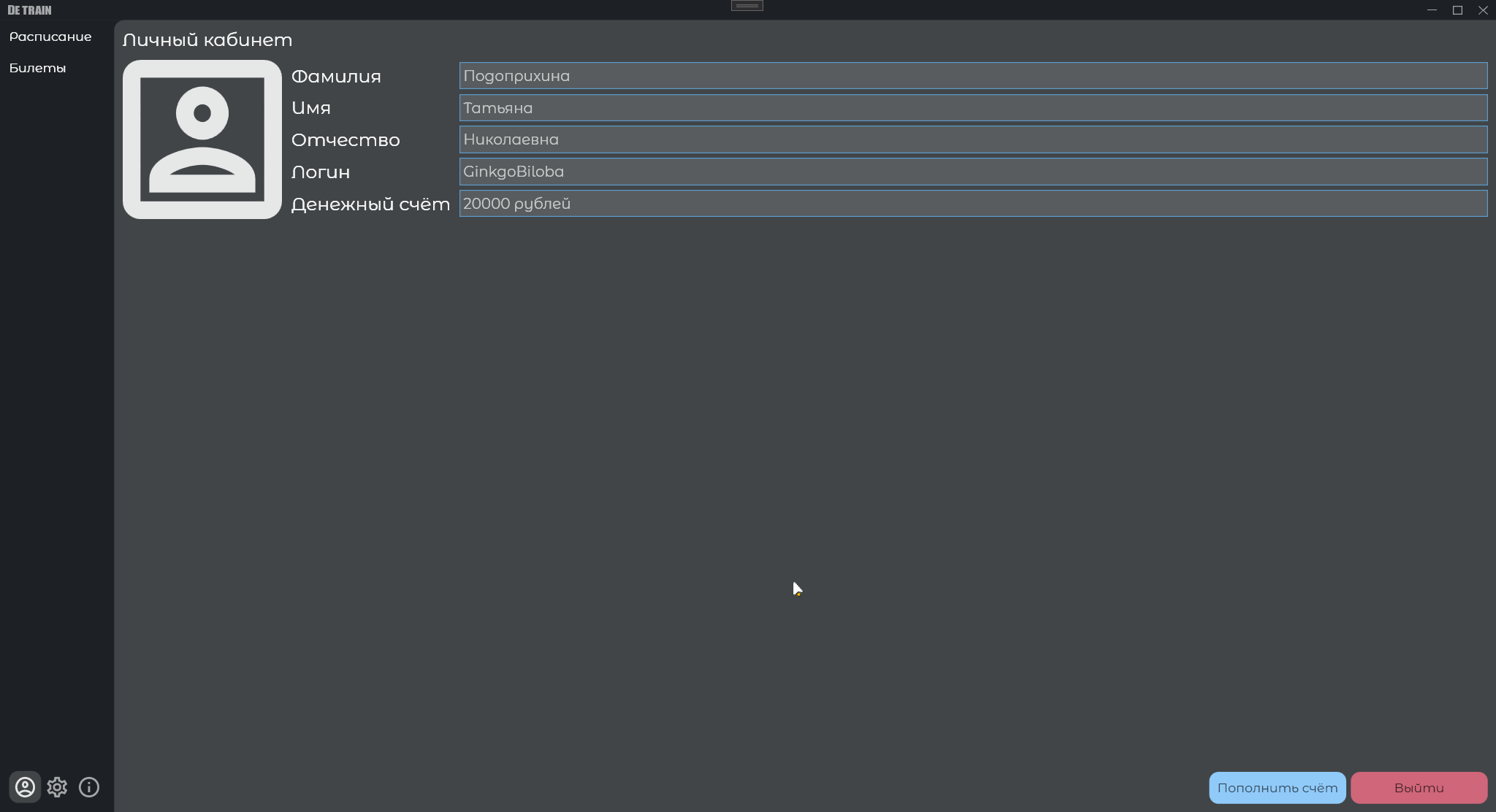


Рисунок 34 – Денежный счёт

# Заключение

В результате выполнения курсового проекта были закреплены теоретические знания и получены практические навыки по созданию CRUD-приложений по реализации баз данных.

А также навыки по проектированию баз данных с помощью таких моделей как «модель использования», «концептуальная модель», «логическая модель» и «физическая модель».

Получен навык локализации крупных приложений, использующих множество типов информации.

Помимо всего прочего был получен опыт коллективной работы в команде разработчиков, с помощью средств удалённого хранения репозиториев «GitHub».

В курсовом проекте полностью решены поставленные задачи.

# Список литературы

1 Васильев А.Н. Программирование на C#. Основные сведения / А.Н. Васильев. — М.: Эксмо, 2018. – 592 с.

2 [Байдачный С. С.](https://biblioclub.ru/index.php?page=author_red&id=77463) .NET Framework 2.0. Секреты создания Windows-приложений: практическое пособие / С. С. [Байдачный.](https://biblioclub.ru/index.php?page=author_red&id=77463) — М.:Солон-Пресс, 2008 – 520 с.

3 Фролов А. В. Визуальное проектирование приложений C# / А. В. Фролов, Г. В. Фролов. —М.: КУДИЦ-Образ, 2003.

4 Сергеева Т.И. Базы данных: модели данных, проектирование, SQL: учебное пособие/ Т.И. Сергеева, М.Ю. Сергеев.: Воронеж ГОУВПО «Воронежский государственный технический университет», 2009. 227 с.

5 Определение паттерна MVVM: <https://metanit.com/sharp/wpf/22.1.php>.

6 Пример реализации паттерна MVVM: <https://metanit.com/sharp/wpf/22.2.php>.