МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

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

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТ:**

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

Тема «Программное средство «Общественный транспорт города»»

Исполнитель

студент 2 курса 4 группы Наварич Андрей Евгеньевич

(Ф.И.О.)

Руководитель работы ассистент Чистякова Ю.А (учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2023

**Содержание**

[Введение 4](#_Toc135493675)

[1. Аналитический обзор литературы и формирование требований 5](#_Toc135493676)

[1.1. Анализ прототипов 5](#_Toc135493677)

[1.1.1. Kogda.by 5](#_Toc135493678)

[1.1.2. Zippy.bus 5](#_Toc135493679)

[1.1.3. Minsk.btrans.by 6](#_Toc135493680)

[1.1.4. Raspisanie.in 7](#_Toc135493681)

[1.1.5. Реальное табло 7](#_Toc135493682)

[1.2. Требования к проекту 8](#_Toc135493683)

[2. Анализ требований к программному средству и разработка функциональных требований 9](#_Toc135493684)

[2.1. Описание средств разработки 9](#_Toc135493685)

[2.2. Спецификация функциональных требований к программному средству 11](#_Toc135493686)

[2.3. Спецификация функциональных требований 11](#_Toc135493687)

[3. Проектирование программного средства 13](#_Toc135493688)

[3.1. Общая структура 13](#_Toc135493689)

[3.2. Взаимоотношения между классами 15](#_Toc135493690)

[3.3. Модель базы данных 16](#_Toc135493691)

[3.4. Проектирование архитектуры проекта 18](#_Toc135493692)

[3.5. Проектирование последовательностей проекта 18](#_Toc135493693)

[4. Реализация программного средства 19](#_Toc135493694)

[4.1. Основные классы программного средства 19](#_Toc135493695)

[4.2. Описание классов и методов программного средства 19](#_Toc135493696)

[4.2.1. Выполнение входа 19](#_Toc135493697)

[4.2.2. Регистрация 19](#_Toc135493698)

[4.2.3. Выбор варианта поиска 20](#_Toc135493699)

[4.2.4. Просмотр Расписания 20](#_Toc135493700)

[4.2.5. Добавление в избранное 20](#_Toc135493701)

[4.2.6. Избранное 21](#_Toc135493702)

[4.2.7. Выход из приложения 21](#_Toc135493703)

[4.2.8. Панель администратора 21](#_Toc135493704)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 22](#_Toc135493705)

[5.1. Тестирование авторизации и регистрации 22](#_Toc135493706)

[5.2. Тестирование административной части 23](#_Toc135493707)

[6. Руководство по установке и использованию 25](#_Toc135493708)

[Заключение 31](#_Toc135493709)

[Список литературы 32](#_Toc135493710)

[ПРИЛОЖЕНИЕ А 32](#_Toc135493711)

[ПРИЛОЖЕНИЕ Б 33](#_Toc135493712)

[ПРИЛОЖЕНИЕ В 34](#_Toc135493713)

[ПРИЛОЖЕНИЕ Г 35](#_Toc135493714)

## **Введение**

Программное средство «Общественный транспорт города» представляет собой комплексный набор сведений о городских транспортных маршрутах. Такое приложение необходимо людям, которые зачастую перемещаются на общественном транспорте и нуждаются в актуальной информации о графике и движении транспорта. С помощью такого ПО можно легко узнать расписание движения автобусов, трамваев, маршруток, поездов и других видов транспорта, а также узнать информацию об изменении маршрута, задержке или отмене.

Более того, ПО обеспечивает возможность отслеживания положения транспорта в режиме реального времени. Это позволяет найти наиболее удобный и быстрый маршрут, избежать пробок, узнать, когда точно прибудет нужный транспорт на остановку.

Приложение так же будет содержать функции личного кабинета, что позволит добавлять необходимые маршруты или остановки в избранное и, благодаря этому, удобнее ориентироваться в расписании. Так же удобство обеспечивает поиск по фильтрам.

Таким образом приложение является актуальным в современном мире, это удобный и необходимы инструмент для тех, кто часто перемещается на транспорте, зачастую студентов. Оно экономит время и силы затраченный на ориентирование в городском транспорте.

## **Аналитический обзор литературы и формирование требований**

## **Анализ прототипов**

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

# **Kogda.by**

Наиболее популярным сервисом по просмотру расписания в интернете является сайт kogda.by. На это сайте представлены возможности выбора вида транспорта, регистрации и впоследствии добавления нужных маршрутов в избранное, так же удобно реализовано табло, которое показывает ближайшие маршруты или, по желанию, полное.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

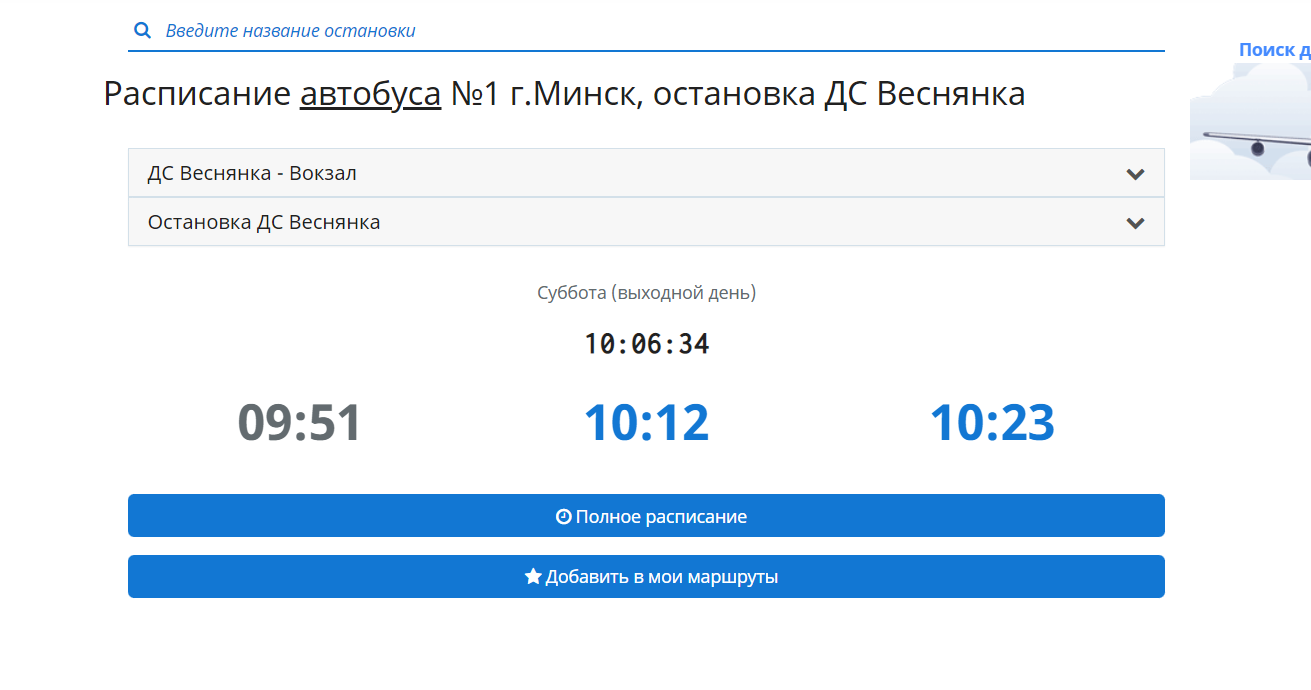


Рисунок 1.1 – Интернет-ресурс «Kogda.by»

# **Zippy.bus**

Еще одним альтернативным решением задач, поставленных в данном курсовом проекте, является интернет-ресурс «Zippy.Bus». С помощью данного программного средства пользователь может ознакомиться с расписанием необходимого ему транспорта, вне зависимости от его вида. Минусом данного ресурса является его дизайн и вид электронного табло.

Одним из главных недостатков данного сайта является отсутствие регистрации и, соответственно, возможности реализации избранного, клиента администратора и прочих преимуществ регистрации.

Интерфейс интернет-ресурса представлен на рисунке 1.2.

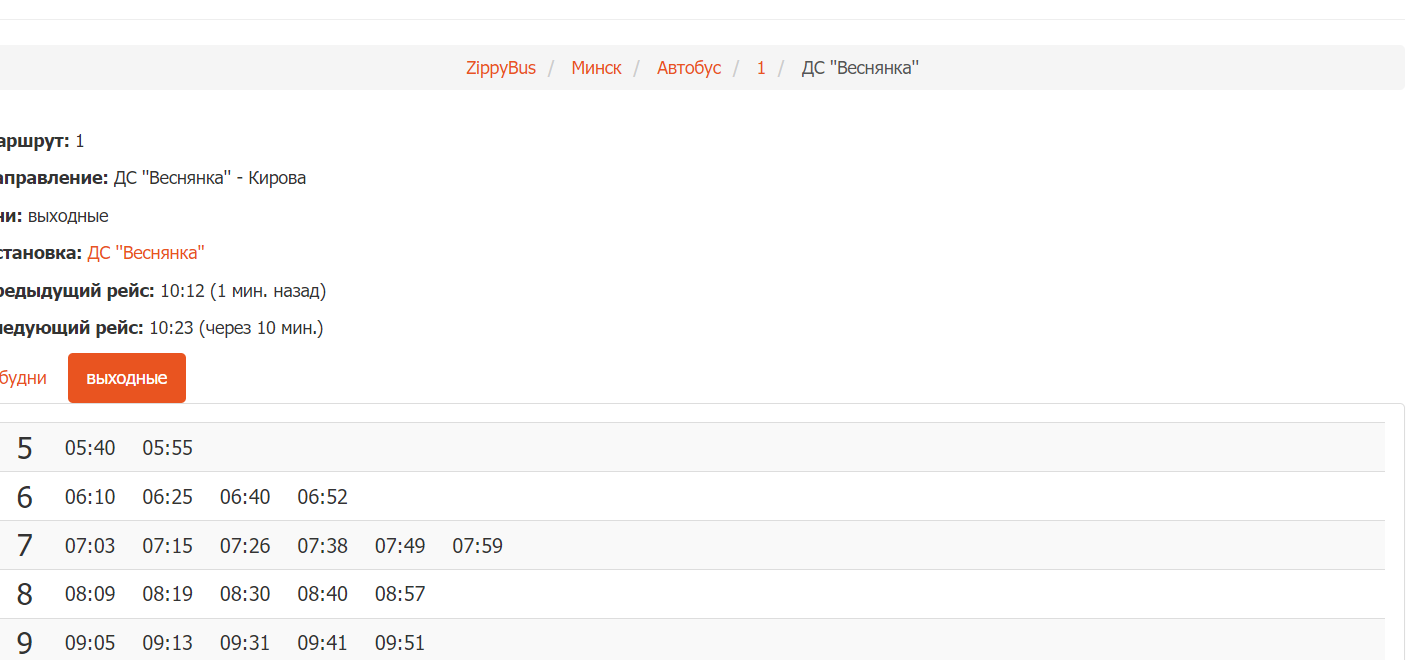


Рисунок 1.2 – Интернет-ресурс «Zippy.bus»

# **Minsk.btrans.by**

Третьей альтернативой решения поставленных в курсовом проекте задач является ресурс «Minsk.btrans.by». Данный сайт так же имеет недостаток в дизайне, в частности реализация электронного табло не является удобной для понимая для большей части пользователей, однако реализация ближайших маршрутов – очень удобная функция для восприятия.

Интерфейс интернет-ресурса представлен на рисунке 1.3.

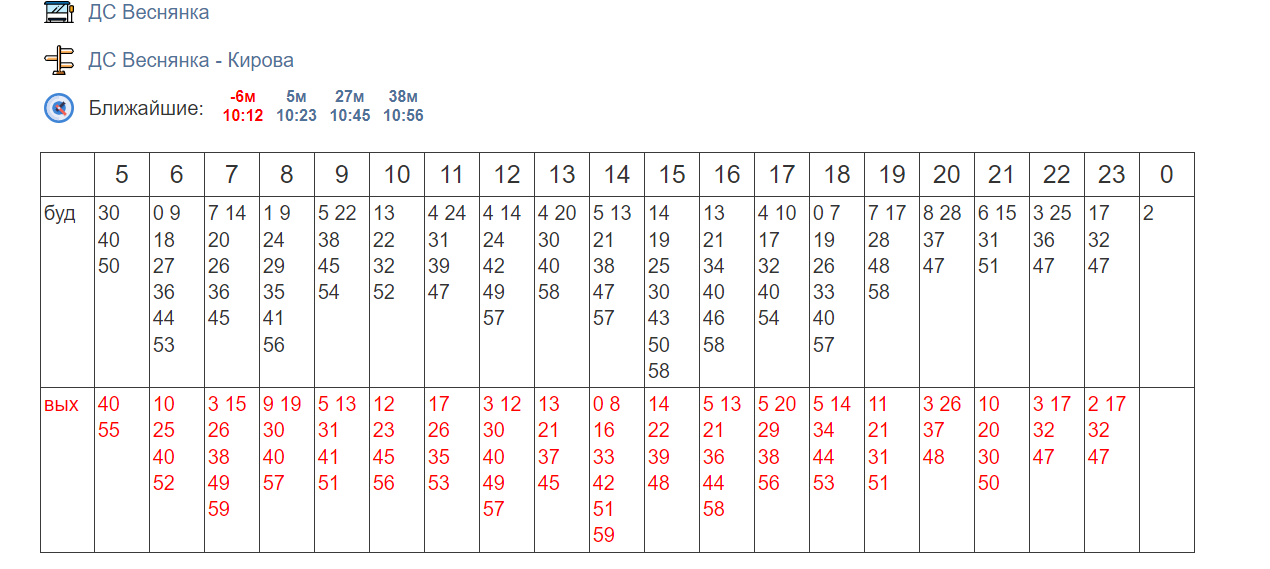


Рисунок 1.3 – Интернет-ресурс «Misk.btrans.by»

Так же, вновь, наблюдается отсутствие функции регистрации, как было сказано выше, это серьезный недостаток из-за ограничения удобного функционала данной темы.

# **Raspisanie.in**

В список рассматриваемых альтернатив решения поставленных задач также можно включить интернет-ресурс «Raspisanie.in». Данный ресурс является практически полной копией ресурса «Mins.btrans.by». Соответственно минусы и плюсы у данных ресурсов идентичные, т.е. отсутствие понятия «пользователь» и другие сопутствующие этому проблемы.

Интерфейс интернет-ресурса представлен на рисунке 1.4.

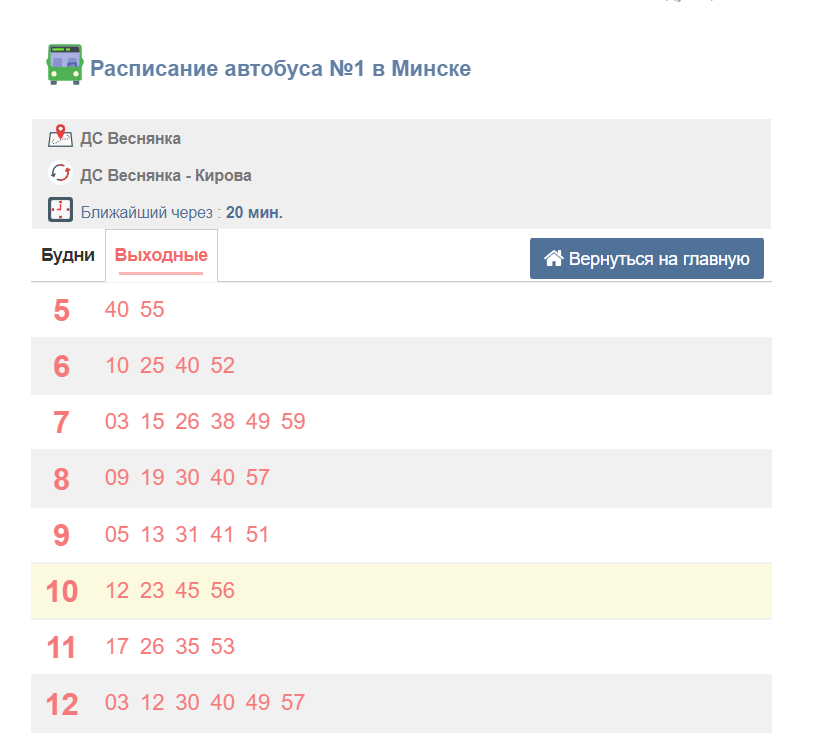


Рисунок 1.4 – Интернет-ресурс «Raspisanie.in»

# **Реальное табло**

Также альтернативой решения поставленных задач является просмотр реального расписания на любом остановочном пункте. На современных остановках расположены электронные табло, которые могут показывать только ближайший и следующий за ним транспорт. Минусом такого подхода является отсутствие возможности планирования своих действий, в свою очередь они ускоряют поиск ближайшего маршрута в отличие от стандартных бумажным видов табло.

## **Требования к проекту**

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

* управление администратором базой данных;
* возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* возможность авторизированным пользователям просмотра информации и её добавления в избранное;
* возможность удобного выбора необходимого транспорта различными путями.

## **Анализ требований к программному средству и разработка функциональных требований**

# **Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2019;
* программная платформа .NET Framework 3.1;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework Core;
* паттерн Command;
* паттерн MVVM;
* MS SQL Server.

1. **Microsoft Visual Studio 2022**

Microsoft Visual Studio 2019 — это интегрированная среда разработки для написания, отладки и сборки кода, а также последующей публикации приложений. Данный продукт позволяет разрабатывать не только консольные, но и десктопные приложения, с использованиям таких технологий, как WinForms или WPF.

1. **Программная платформа .NET Framework 3.1**

Платформа .NET Framework — это созданная в 2002 году технология Microsoft, которая поддерживает создание и выполнение веб-служб и приложений Windows, обеспечивающая согласованную объектно-ориентированной среду программирования для локального сохранения и выполнения объектного кода, локального выполнения кода, распределенного в Интернете, либо для его удаленного выполнения. Основой платформы является общеязыковая среда Common Language Runtime, выполняющая код на разных поддерживаемых языках программирования.

1. **Язык программирования C#**

В качестве языка программирования используется C# – основной язык разработки в .NET Framework.

Язык объектно-ориентированный, имеет строгую статическую типизацию, поддерживает перегрузку операторов, указатели на функции-члены классов, атрибуты, события, свойства, исключения.

Используется как основной язык в технологии WPF.

1. **Технология WPF**

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

1. **Расширяемый язык разметки XAML**

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

1. **Технология Entity Framework Core**

Для взаимодействия с базой данных в приложении WPF используется технология Entity Framework Core — кроссплатформерная и расширяемая версия технологии Entity Framework с открытым исходным кодом, которая в текущий момент развивается и дополняется.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Entity Framework Core – это последняя версия Entity Framework 7, которая совместима с .NET Core и на данный момент продолжает развиваться. Entity Framework Core предполагает три возможных способа взаимодействия с базой данных:

* Database First: Entity Framework Core создает набор классов, которые отражают модель конкретной базы данных;
* Model First: сначала разработчик создает модель базы данных, по которой затем Entity Framework Core создает реальную базу данных на сервере;
* Code First: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework Core по этой модели генерирует базу данных и ее таблицы.

1. **Паттерн Command**

Технология Command — это поведенческий паттерн проектирования, который позволяет инкапсулировать запрос на выполнение какой-то операции в виде отдельного объекта. Этот объект может быть передан другому объекту или вызван позднее.

Благодаря этому, мы можем разделить клиентский код, который инициирует запрос на выполнение операции, от кода, который фактически выполняет эту операцию. Это помогает упростить код, улучшить его масштабируемость и обеспечить более гибкую архитектуру программной системы.

1. **MS SQL Server**

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

# **Спецификация функциональных требований к программному средству**

Программное средство должно предоставлять следующие функциональные возможности:

Для пользователя:

* регистрация;
* авторизация;
* просмотр маршрутов по номеру;
* просмотр маршрутов по остановке;
* добавление в избранное;

Для администратора:

* авторизация;
* изменение временного расписания маршрута;
* изменение составляющих маршрута;
* изменение свойств транспорта.

# **Спецификация функциональных требований**

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

В программном средстве необходимо реализовать регистрацию и авторизацию пользователей для доступа ко всем возможностям приложения. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Для регистрации входными данными являются имя, фамилия, логин, пароль, e-mail и номер телефона. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных.

Пользователь может просматривать расписание, производить поиск и фильтрацию и пользоваться избранным. Для каждого пользователя создается его профиль, в котором он может просмотреть информацию о добавленных маршрутах. Все сведения пользователе должны автоматически обновляться в базе данных.

Администратор не обладает теми же возможностями, он может работать только с базой данных и просматривать стандартные маршруты, функция избранного у администратора отключена.

## **Проектирование программного средства**

# **Общая структура**

Программное средство «AllTrans» имеет следующую структуру, представленную на рисунке 3.1.

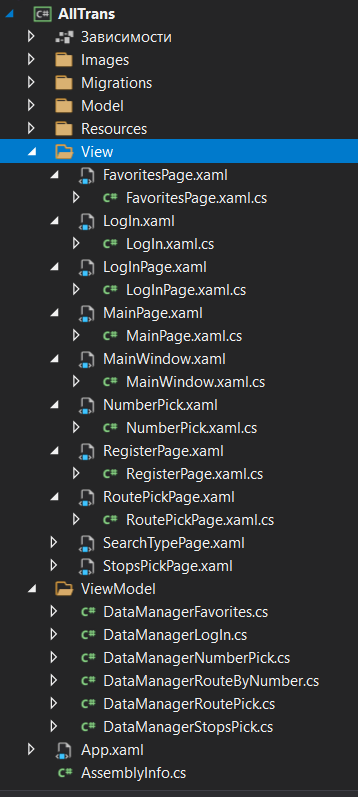


Рисунок 3.1 – Структура проекта

Как можно заметить проект состоит из объектов скомпонованых по принципам паттерна MVVM.

Так же существуют папки с реализацией сущностей EntityFramework.

Файлы ресурсов и изображений, которые подгружаются в проект из вне.

Файлы конфигурации приложения хранятся отдельно вне папок.

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

Таблица 3.1 – Описание структуры папок и файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| App.xaml | Файл с параметрами проекта. |
| Папка Model | Папка, содержащая основные классы-модели проекта. |
| Папка Resources | Папка, содержащая все словари ресурсов. |
| Папка Images | Папка, содержащая все используемые изображения. |
| Папка ViewModel | Папка, содержащая все модели View проекта. |
| DataManagerFavorites.cs | Класс-модель для работы со страницей-избранным. |
| DataManagerLogIn.cs | Класс-модель для работы со страницей-авторизацией. |
| DataManagerNumberPick.cs | Класс-модель для работы со страницей-расписанием. |
| DataManagerRouteByNumber.cs | Класс-модель для работы со страницей для отображения маршрутов по номеру. |
| DataManagerRoutePick.cs | Класс-модель для работы со страницей для отображения маршрутов. |
| DataManagerStopsPick.cs | Класс-модель для работы со страницей для выбора остановок |
| Папка View | Папка, содержащая объекты визуального отображения проекта View. |
| NumberPick.xaml | Страница для выбора номерат транспорта. |
| StopsPickPage.xaml | Окно для выбора остановки. |
| RoutesPickPage.xaml | Страница с получением информации о маршруте. |
| SearchTypePage.xaml | Страница для выбора филтрации. |
| LogIn.xaml | Окно для выполнения входа. |
| LogInPage.xaml | Страница для выполнения входа |
| MainPage.xaml | Основная страница приложения. |
| MainWindow.xaml | Основное окно приложения. |
| FavoritesPage.xaml | Страница для избранного. |
| RegistrationPage.xaml | Страница для регистрации. |

Более подробная структура содержимого папок программного средства описана на рисунке 3.2 и в таблице 3.2.

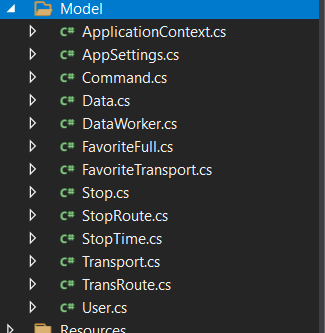


Рисунок 3.2 – Подробная структура проекта

Таблица 3.2 – Подробное описание структуры файлов проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| ApplicationContext.cs | Вспомогательный класс для работы с EntityFramework. |
| AppSettings.cs | Вспомогательный класс для хранения данных о пользователе. |
| Command.cs | Класс реализующий паттерн - Command. |
| Data.cs | Класс содержащий статические данные. |
| DataWorker.cs | Класс реализующий функции Entity с базой данных. |
| FavoriteFull.cs | Класссодержащий информацию об избраннном. |
| FavoriteTransport.cs | Класс сущности FavoriteTransport. |
| Stop.cs | Класс сущности Stop. |
| StopRoute.cs | Класс сущности StopRoute. |
| StopTime.cs | Класс сущности StopTime. |
| Transport.cs | Класс сущности Transport. |
| TransRoute.cs | Класс сущности TransRoute. |
| User.cs | Класс сущности User. |

# **Взаимоотношения между классами**

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

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

# **Модель базы данных**

Для реализации поставленной задачи была создана база данных KursovoiProektBD. Для ее создания использовалась система управления реляционными базами данных MSQL Server. База данных состоит из таблиц, представленных на рисунке 3.3. Скрипт для создания базы данных представлен в приложении Г.

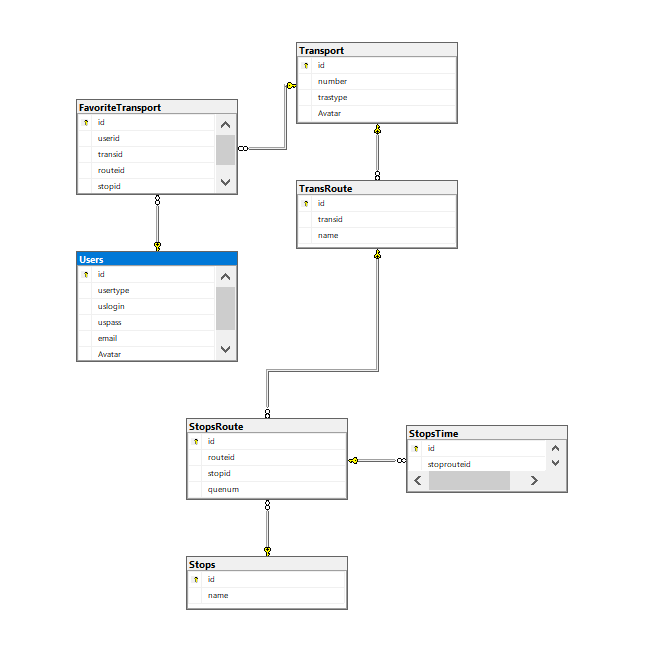


Рисунок 3.3 – База данных KursovoiProektBD

На рисунке 3.4 проиллюстрирована структура таблицы «Users», которая содержит информацию о пользователях. В данной таблице поле id, представленное int, является первичным ключом. Поле uslogin хранит логин пользователя, uspass– пароль, email– адрес электронной почты, Avatar и usertype – соответственн аватар и роль (admin для администратора, user для пользователя).

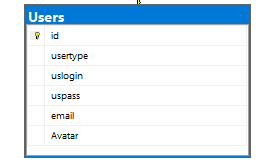


Рисунок 3.4 – Структура таблицы «Users»

На рисунке 3.5 изображена структура таблицы «Stops», содержащая информацию об остановках. Первичный ключ ID хранит уникальный идентификатор остановки, name – название.

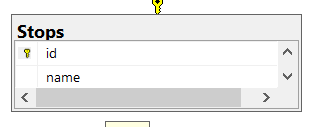


Рисунок 3.5 – Структура таблицы «Stops»

На рисунке 3.6 показана структура таблицы «Transport», содержащая данные о совершённых заказах билетов. В данной таблице первичный ключ ID является идентификатором транспорта, поле number ­­­­­­­­­– номер автобуса, trastype – вид транспорта, Avatar – изображение автобуса.

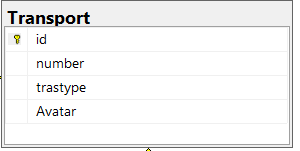


Рисунок 3.6 – Структура таблицы «Transport»

На рисунке 3.7 показана структура таблицы «TransRoute», содержащая данные о совершённых заказах билетов. В данной таблице первичный ключ ID является идентификатором транспорта, поле name­­­­­­­­­ – имя маршрута, transid– id связанного с маршрутом транспорта.

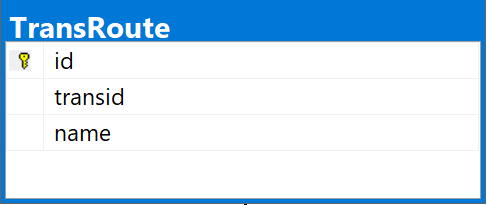


Рисунок 3.7 – Структура таблицы «TransRoute»

Другие таблицы базы данных являются вспомогательными и составными из полей таблиц, описанных выше.

# **Проектирование архитектуры проекта**

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

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

Диаграмма использования представлена в приложении Б.

# **Проектирование последовательностей проекта**

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

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

Диаграмма последовательностей представлена в приложении В.

## **Реализация программного средства**

# **Основные классы программного средства**

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

* Выполнение входа;
* Регистрация;
* Просмотр маршрутов по остановке;
* Просмотр маршрутов по номеру;
* Добавление в избранное;
* Просмотр расписания из избранного;
* Функционирование администратора;
* Выполнение выхода.

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

# **Описание классов и методов программного средства**

## **Выполнение входа**

Необходимый функционал для возможности входа пользователя в аккаунт, а также для проверки корректности данных, реализован в классе DataManagerLogIn.

Основным методом класса является метод LoginCommand, выполняющийся при нажатия пользователя на кнопку «Войти». В данном методе вызывается метод GetUser(), ищущий введённые значения в базе данных пользователей.

При успешной валидации введённых пользователем входных данных выполняется проверка на существование введённого логина в базе данных, и при нахождении значения выполняется проверка на соответствие введённого пароля и хранящегося в базе данных пароля. При успешном прохождении всех проверок пользователь перенаправляется на главную страницу, а противном случае – получает сообщение о соответствующей ошибке.

Для перехода к странице регистрации со страницы входа реализован метод LogEvent(), выполняющийся при нажатии пользователя на кнопку «Регистрация» на странице входа, и перенаправляющий пользователя со страницы входа на страницу регистрации.

Листинг реализации класса DataManagerLogIn представлен в приложении Г.

## **Регистрация**

Для совершения пользователем регистрации нового аккаунта реализован класс DataManagerLogIn.

При нажатии пользователя на кнопку «Зарегистрироваться» вызывается основной метод класса – RegisterCommand(). В начале все введённые пользователем данные проходят проверку.

После успешного прохождения вышеописанных проверок происходит попытка добавить пользователя в базу данных и перенаправление пользователя на страницу входа. Если логин, адрес электронной почты или номер телефона уже зарегистрирован в базе данных, то пользователь получает соответствующее уведомление об ошибке.

Листинг реализации класса LogIn представлен в приложении Г.

## **Выбор варианта поиска**

После успешной авторизации пользователь перенаправляется на основную страницу программного средства, представленную классом MainWindow, позволяющую просматривать и фильтровать все представленные фильмы.

Реализованный функционал главной страницы позволяет пользователю производить фильтрацию по остановке или по номеру, отражёнными в соответственных методах классов DataManagerStopsPick и DataManagerNumberPick.

Первый из классов в зависимости от значения выбранного транспорта выводит список изображений транспорта соответствующего типа в элемент ListView. При клике на номер транспорта производится перевод на страницу с выбором маршрута и остановки, которым принадлежит транспорт. Затем можно просмотреть ближайшее, дальнее и прошлое время прибытия, добавить маршрут в избранное при нажатии кнопки, а так же просмотреть полное расписание в виде списка.

Второй класс DataManagerStopsPick реализует поиск по остановке. Для начала появляется окно, вызывающее окно StopsPickPage.xaml. В этом окне содержатся ComboBox, в которых содержатся Маршруты, при выборе которого появится возможность выбора остановки. Затем при помощи класса DataManagerRoutePick реализуется вывод в список ListView изображений транспорта подходящих под выбранный фильтр

Листинги реализации класса DataManagerStopsPick, DataManagerRoutePick и DataManagerNimberPick представлен в приложении Г.

## **Просмотр Расписания**

При клике на номер транспорта производится перевод на страницу с выбором маршрута и остановки, которым принадлежит транспорт. Затем можно просмотреть ближайшее, дальнее и прошлое время прибытия, добавить маршрут в избранное при нажатии кнопки, а так же просмотреть полное расписание в виде списка. Так как мы заранее выбрали маршрут и остановку, в конструктор функций класса DataManagerRoutePick мы уже передаём номер, маршрут и остановку затем мы получаем такой же функционал, что и при просмотре по номеру.

Листинг реализации класса ReserveTicket представлен в приложении Г.

## **Добавление в избранное**

После вышеописанного выбора пользователь может добавить любой элемент в избранное благодаря классу DataManagerFavorites. Этот класс пользуясь коллекциями класса FavoriteTrans и FavoriteFull, добавляет в избранное выбранный транспорт благодаря методу Fill();

Листинг реализации класса DataManagerFavorites представлен в приложении Г.

## **Избранное**

Персональная страница пользователя, которая реализуется классом FavoriteTrans и DataManagerFavorites. При нажатии на кнопку «Звезда» производится переход на страницу, благодаря методу FavoritesButton. В окне избранного пользователь может увидть ListView со списком маршрутов, состоящий из номера, маршрута и остановки, а так же кнопки «полное расписание». При нажатии кнопки полное расписание производится переход на страницу RoutesPickPage где уже установлены стандартные настройки данной страницы.

Листинг реализации класса DataManagerFavorites представлен в приложении Г.

## **Выход из приложения**

Выход из приложения производится методом LogOutClick окна MainWindow. При выходе пользователь переносится на окно входа/регистрации.

## **Панель администратора**

Администратор обладает функцией изменения времени маршрута, данный интерфейс представлен в классе DataManageRoutePickPage методом ChangeTime(), при выборе жлемента из полного списка и нажатии кнопки «Изменить время», которая появляется при авторизации, как администратор, при это администратор не обладает функцией избранного, что регулируется в классе MainWindow

Листинг реализации класса DataManageRoutePickPage представлен в приложении Г.

## **Тестирование, проверка работоспособности и анализ полученных результатов**

## **Тестирование авторизации и регистрации**

В момент авторизации и регистрации возможна ситуация, когда пользователь вводит некорректные данные, например, неверный пароль, незарегистрированный логин, некорректный адрес электронной почты. Такие исключения обрабатываются программным средством с помощью показа пользователю всплывающих сообщений с текстом ошибки.

Пример ошибки неверного логина приведён на Рисунке 5.2.

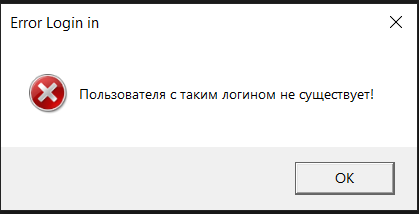


Рис 5.1 – Обработка неверного логина

Обработка неверного пароля приведена на рисунке 5.2.

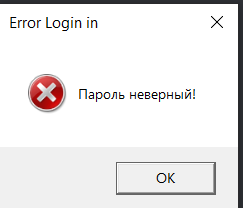


Рис 5.2 – Обработка неверного пароля

Обработка неверного ввода логина при регистрации приведена на Рисунке 5.3.

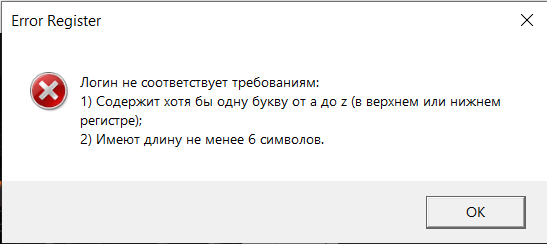


Рис 5.3 – Обработка неверного ввода логина

Обработка неверного email представлена на рисунке 5.4.

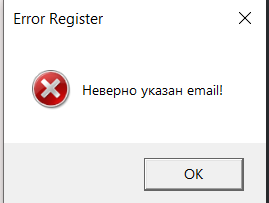


Рис 5.4 – Обработка ввода несуществующего логина

Таким образом были реализованы стандартные функции проверки при авторизации и регистрации пользователя.

## **Тестирование административной части**

При использовании формы будучи авторизированным как администратор накладывается ограничение на доступ к избранному. Пример такого случая приведён на рисунке 5.5.

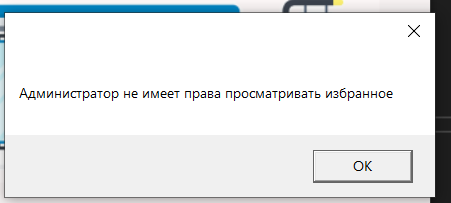


Рис 5.5 – Сообщение о входе в избранное от админа

Таким образом были протестированы все возможные варианты проверок и запретов в приложении.

## **Руководство по установке и использованию**

При запуске программного средства «AllTrans» пользователь попадает на страницу авторизации, содержащая формы входа и регистрации.

Стартовая страница позволяет зарегистрированному пользователю совершить вход в аккаунт, а не зарегистрированному – перейти на страницу регистрации нажатием на кнопку «Регистрация» и зарегистрироваться в системе, после чего пользователь будет перенаправлен на страницу авторизации, с которой он может совершить вход в систему.

По умолчанию новому зарегистрированному клиенту присваивается роль «Пользователь». Роль «Администратор» при регистрации не присваивается. Окно авторизации представлено на рисунках 6.1, 6.2.

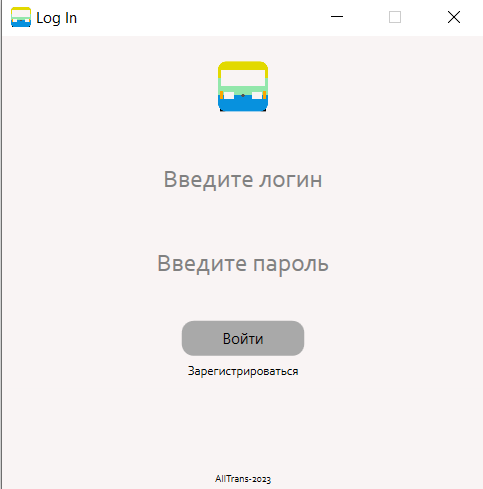


Рисунок 6.1 – Страница авторизации

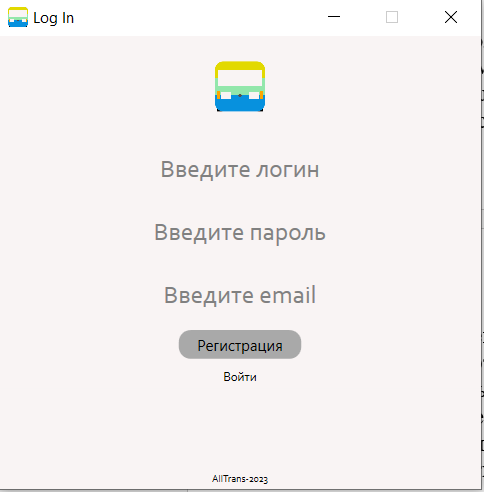


Рисунок 6.2 – Страница регистрации

После выполнения авторизации пользователь перенаправляется на основную страницу программного средства.

Пользователю доступен выбор вида транспорта и вид фильтрации поиска.

Также пользователь имеет возможность попасть в избранное, скрыть навигационную панель и выйти из аккаунта.

Для осуществления поиска необходимо выбрать его вид, и затем в зависимости от выбранного типа настроить фильтры.

Основная страница изображена на рисунке 6.3.

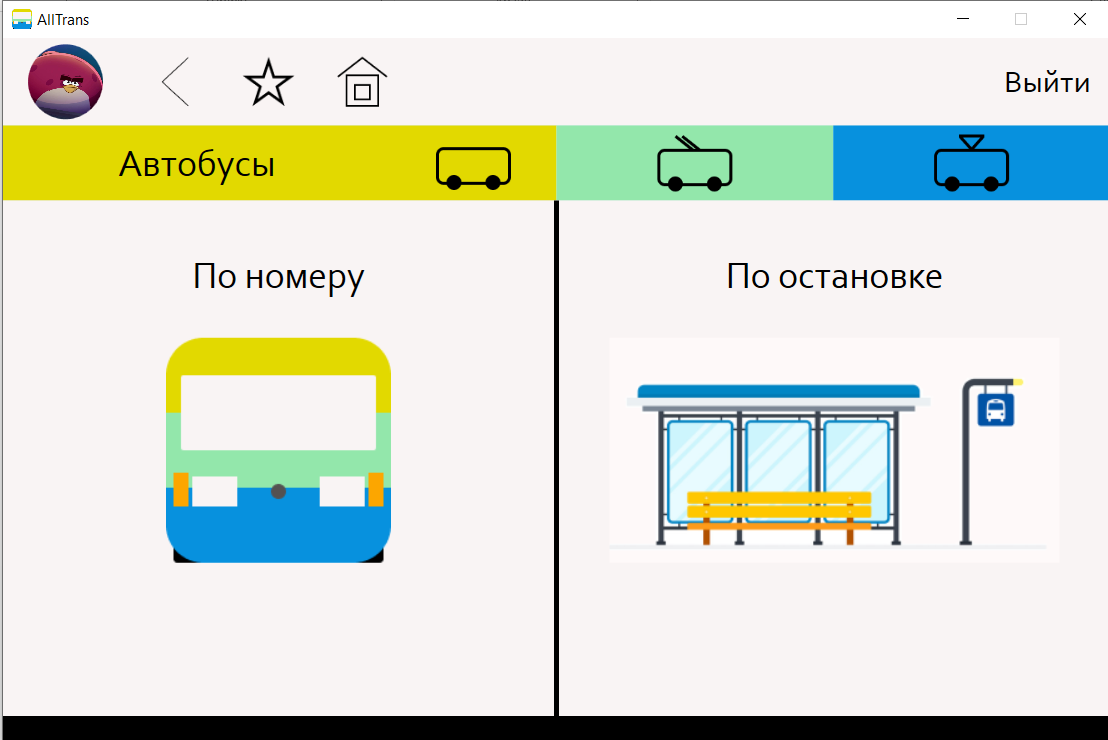


Рисунок 6.3 – Основная страница

При нажатии на кнопку «По номеру» пользователь перенаправляется на страницу с выбором номера маршрута в зависимости от его типа. Для получения дальнейшей информации необходимо выбрать номер. Меню после нажатия «По остановке» практически не отличается от первого варианта, единственное отличие в том, что номера автобусов устанавливаются по фильтрам маршрутов и остановок.

Страница выбора номера представлена на рисунке 6.4.

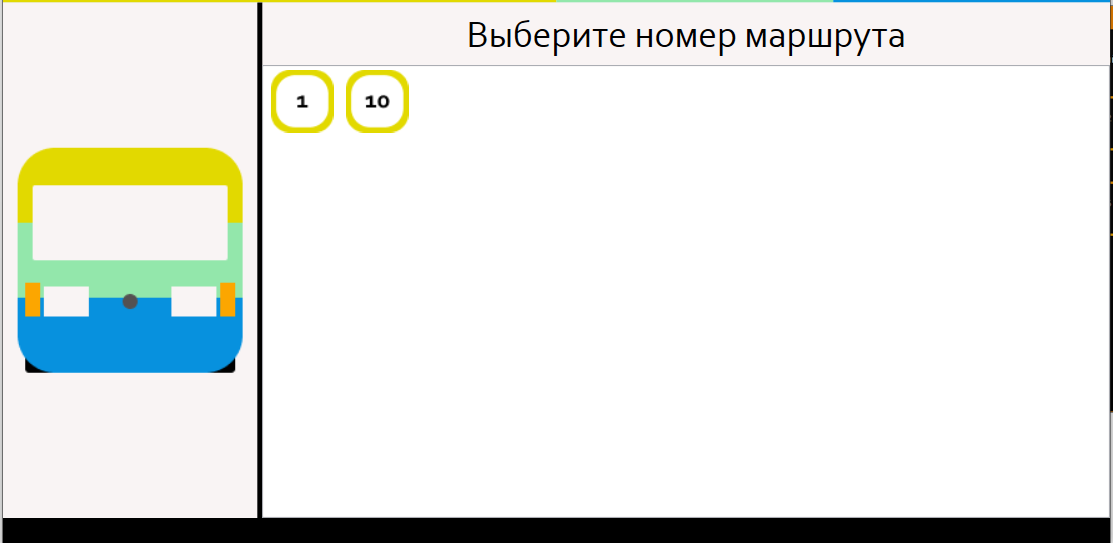


Рисунок 6.5 – Страница выбора номера

После выбора нужного нам номера осуществляется переход на последнюю информационную страницу с расписанием выбранного транспорта, где мы можем просмотреть актуальное расписание, либо по кнопке «Просмотреть полное расписание» просмотреть всё время прибытия. Если маршрут является важным для пользователя, он может добавить его в избранное с помощью нажатия на кнопку «Добавить маршрут в избранное». Страница представлена на рисунке 6.6.

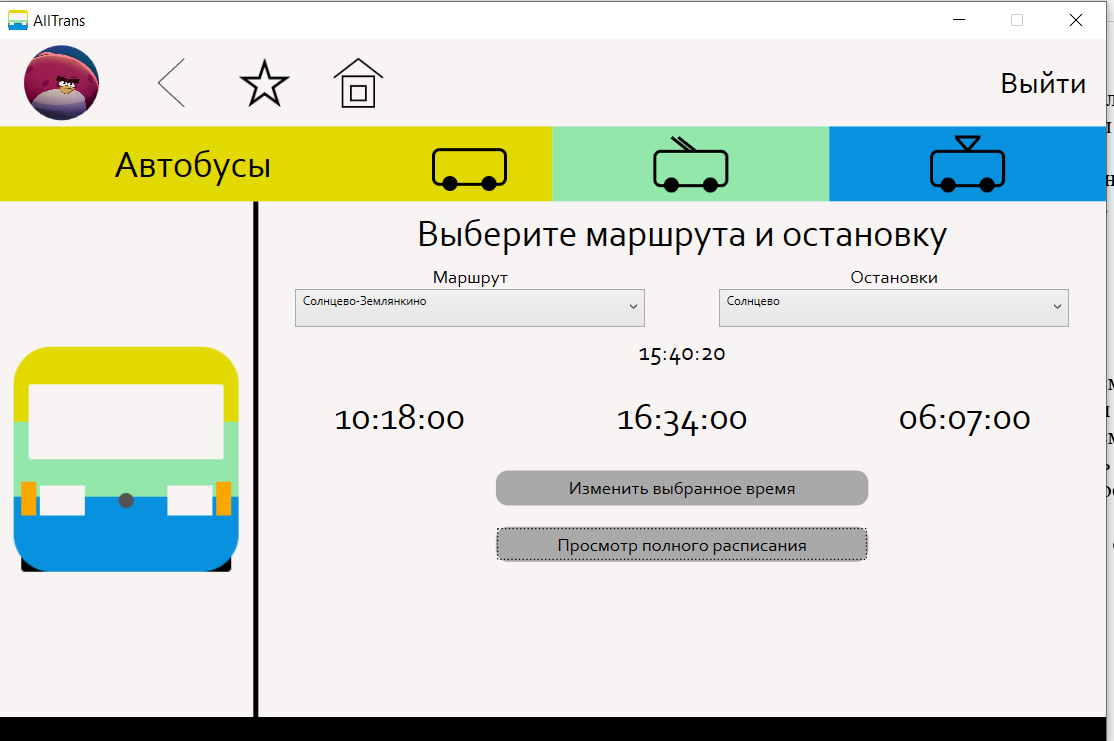


Рисунок 6.6 – Окно просмотра расписания

При надобности пользователь может обратиться в раздел избранное нажав на кнопку в виде звезды, что перенесёт его на новую страницу, состоящую из списка добавленный для каждого пользователя маршрутов в формате «номер-маршрут-остановка», при надобности присутствуют кнопка «полное расписание» при нажатии на которою, происходит перенаправление на страницу просмотра расписания данного маршрута.

Страница избранного представлена на рисунке 6.7.

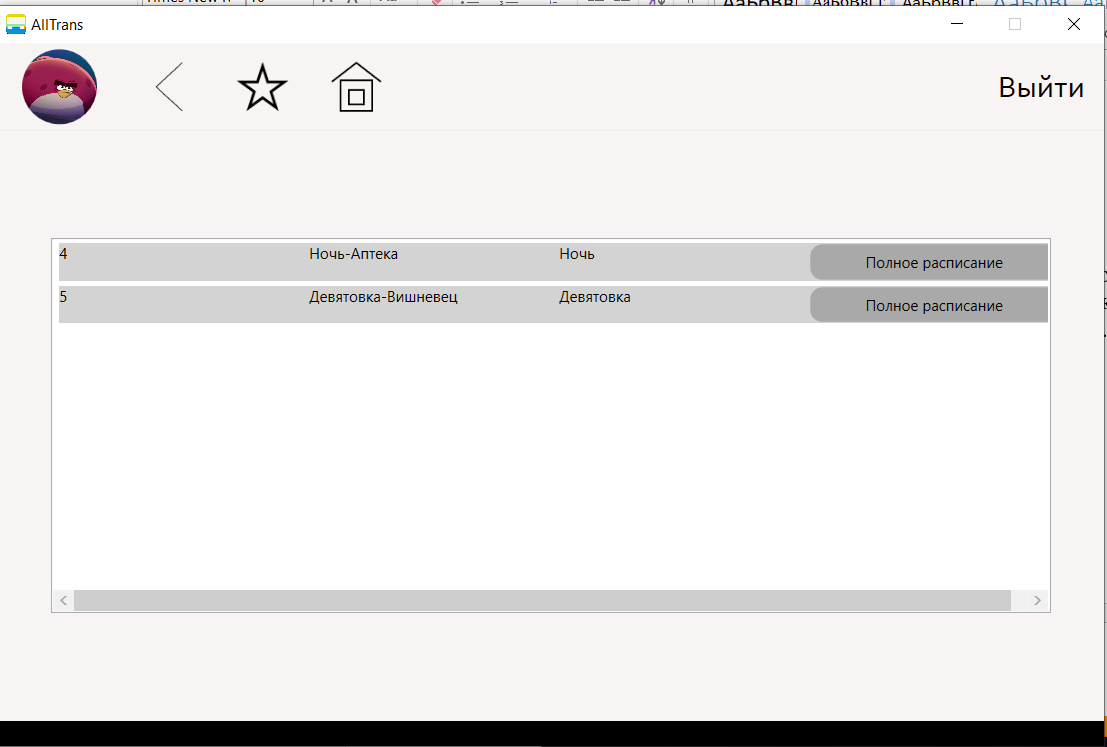


Рисунок 6.7 – Личный кабинет пользователя

Для того чтобы изменить вид транспорта в приложении существуют интерактивные кнопки, меняющие его. Дизайн кнопок представлен на рисунке 6.8.



Рисунок 6.8 – Кнопки выбора типа

Помимо этого, в приложении существует вспомогательный функционал, такой как скрытие навигационного меню и кнопка «Домой».

# **Заключение**

В ходе выполнения курсового проекта было разработано программное средство «AllTrans» на языка C# с использованием технологий Entity Framework Core, WPF.

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

* создана база данных;
* реализована возможность пользователю зарегистрироваться или войти в существующую учетную запись;
* реализована возможность авторизированным пользователям просмотра транспортного расписания;
* реализована возможность просмотра избранного пользователя;
* реализована возможность добавления маршрута в избранное;
* реализовано уведомление пользователя о регистрации.

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

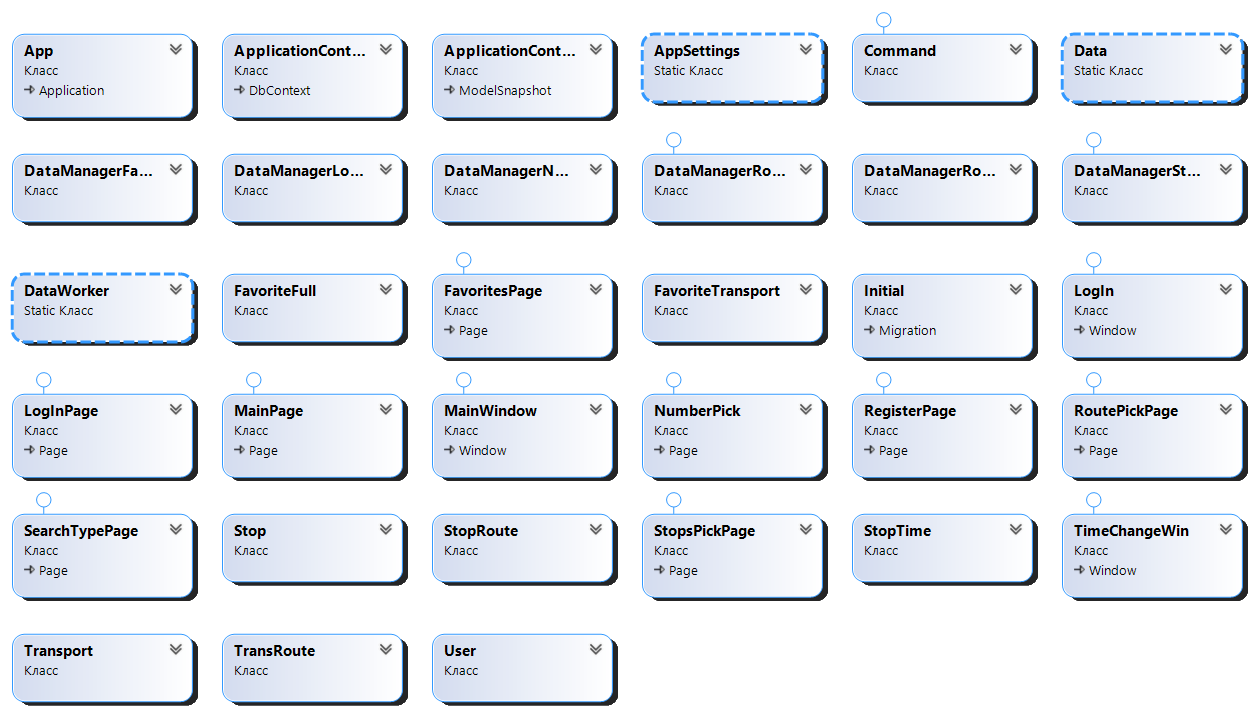
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список литературы**

1. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio – Дата доступа 23.04.2023
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. – Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 23.04.2023
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Режим доступа: Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 25.04.2023
5. Руководство по XAML // [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/xaml/index.htm– Дата доступа: 25.04.2023
6. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.
7. Работа с Entity Framework Core [Электронный ресурс] – Режим доступа: <https://professorweb.ru/my/entity-framework/6/level1/> − Дата доступа 28.04.2023

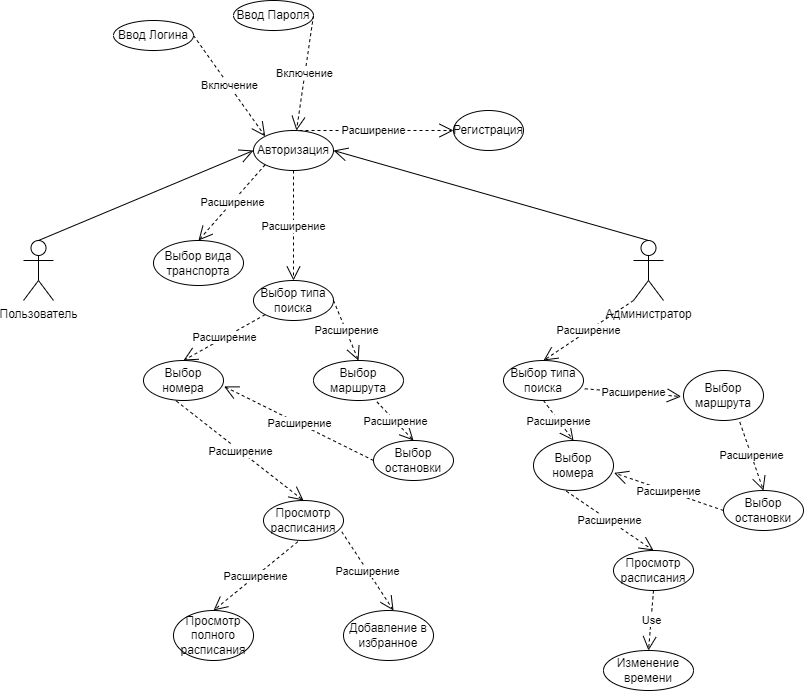
# **ПРИЛОЖЕНИЕ А**

Диаграмма классов



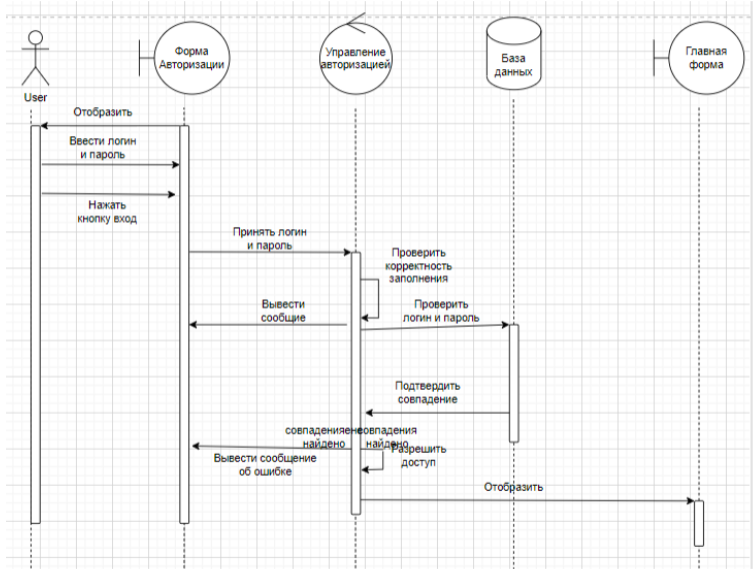
# **ПРИЛОЖЕНИЕ Б**

Диаграмма прецендентов



# **ПРИЛОЖЕНИЕ В**

Диаграмма последовательности



# **ПРИЛОЖЕНИЕ Г**

Скрипт для создания базы данных

create table Transport(

id int not null primary key,

number int not null,

trastype nvarchar(10) not null

)

create table TransRoute(

id int not null primary key,

transid int not null foreign key references Transport(id),

name nvarchar(50) not null)

create table Stops(

id int not null primary key,

name nvarchar(50) not null)

create table StopsRoute(

id int not null primary key,

routeid int not null foreign key references TransRoute(id),

stopid int not null foreign key references Stops(id),

quenum int not null)

create table StopsTime(

id int not null primary key,

stoprouteid int not null foreign key references StopsRoute(id),

timeval time not null)

create table Users(

id int not null primary key,

usertype nvarchar(10) not null,

uslogin nvarchar(10) not null unique,

uspass nvarchar(8) not null,

email nvarchar(50) not null unique)

create table FavoriteStops(

id int not null primary key,

userid int not null foreign key references Users(id),

stopid int not null foreign key references Stops(id))

create table FavoriteTransport(

id int not null primary key,

userid int not null foreign key references Users(id),

transid int not null foreign key references Transport(id))

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

Листинг 1: Класс Login

namespace AllTrans.View

{

/// <summary>

/// Логика взаимодействия для LogIn.xaml

/// </summary>

public partial class LogIn : Window

{

public LogIn()

{

InitializeComponent();

this.LoginFields.Navigate(new LogInPage(this));

}

}

} using AllTrans.ViewModel;

using System;

using System.Collections.Generic;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AllTrans.View

{

/// <summary>

/// Логика взаимодействия для LogInPage.xaml

/// </summary>

public partial class LogInPage : Page

{

public LogIn \_LogIn;

public LogInPage(LogIn logIn)

{

InitializeComponent();

\_LogIn = logIn;

this.DataContext = new DataManagerLogIn();

}

private void LogIn\_TextChanged(object sender, TextChangedEventArgs e)

{

if (LogIn.Text.Length > 0)

{

LogInPlug.Visibility = Visibility.Hidden;

}

else

{

LogInPlug.Visibility = Visibility.Visible;

}

}

private void Password\_TextChanged(object sender, TextChangedEventArgs e)

{

if (PasswordIn.Text.Length > 0)

{

PasswordPlug.Visibility = Visibility.Hidden;

}

else

{

PasswordPlug.Visibility = Visibility.Visible;

}

}

private void RegisterEvent(object sender, MouseButtonEventArgs e)

{

NavigationService.Navigate(new RegisterPage(\_LogIn));

}

}

}

Листинг 2: Класс RegisterPage

using AllTrans.ViewModel;

using System;

using System.Collections.Generic;

using System.Net;

using System.Net.Mail;

using System.Text;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AllTrans.View

{

/// <summary>

/// Логика взаимодействия для RegisterPage.xaml

/// </summary>

public partial class RegisterPage : Page

{

public LogIn log;

public RegisterPage(LogIn logIn)

{

InitializeComponent();

log = logIn;

this.DataContext = new DataManagerLogIn();

}

private void LogIn\_TextChanged(object sender, TextChangedEventArgs e)

{

if (LogIn.Text.Length > 0)

{

LogInPlug.Visibility = Visibility.Hidden;

}

else

{

LogInPlug.Visibility = Visibility.Visible;

}

}

private void Password\_TextChanged(object sender, TextChangedEventArgs e)

{

if (PasswordIn.Text.Length > 0)

{

PasswordPlug.Visibility = Visibility.Hidden;

}

else

{

PasswordPlug.Visibility = Visibility.Visible;

}

}

private void Email\_TextChanged(object sender, TextChangedEventArgs e)

{

if (emailIn.Text.Length > 0)

{

emailPlug.Visibility = Visibility.Hidden;

}

else

{

emailPlug.Visibility = Visibility.Visible;

}

}

private void LogInEvent(object sender, MouseButtonEventArgs e)

{

NavigationService.Navigate(new LogInPage(log));

}

public static MailMessage CreateMail(string name, string emailFrom, string emailTo, string subject, string body)

{

var from = new MailAddress(emailFrom, name);

var to = new MailAddress(emailTo);

var mail = new MailMessage(from, to);

mail.Subject = subject;

mail.Body = body;

mail.IsBodyHtml = true;

return mail;

}

public static void SendMail(string host, int smptPort, string emailFrom, string pass, MailMessage mail)

{

SmtpClient smtp = new SmtpClient(host, smptPort);

smtp.Credentials = new NetworkCredential(emailFrom, pass);

smtp.EnableSsl = true;

smtp.Send(mail);

}

}

}

Листинг 3: Класс MainWindow

using AllTrans.Model;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace AllTrans.View

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public bool NavStat = true;

private MainPage mainPage = new MainPage();

private LogIn logIn = new LogIn();

public MainWindow()

{

InitializeComponent();

ContentContainer.Navigate(mainPage);

}

private void CoverNav(object sender, MouseButtonEventArgs e)

{

if(NavStat)

{

this.NavPanel.ColumnDefinitions[2].Width = new GridLength(0);

this.NavPanel.ColumnDefinitions[3].Width = new GridLength(0);

this.CoverImage.Source = new BitmapImage(new Uri(@"\Images\Arrow.png", UriKind.Relative));

NavStat = false;

}

else

{

this.NavPanel.ColumnDefinitions[2].Width = new GridLength(75);

this.NavPanel.ColumnDefinitions[3].Width = new GridLength(75);

this.CoverImage.Source = new BitmapImage(new Uri(@"\Images\ArrowBack.png", UriKind.Relative));

NavStat = true;

}

}

private void HomeButton(object sender, MouseButtonEventArgs e)

{

ContentContainer.Navigate(new MainPage());

mainPage.SearchTypeFrame.Navigate(new View.SearchTypePage());

Model.Data.NumberListStatus = false;

Model.Data.StopListStatus = false;

}

private void FavoritesNav(object sender, MouseButtonEventArgs e)

{

if(Data.UserIsAdmin)

{

MessageBox.Show("Администратор не имеет права просматривать избранное");

}

else

{

ContentContainer.Navigate(new FavoritesPage());

Model.Data.NumberListStatus = false;

Model.Data.StopListStatus = false;

}

}

private void LogEvent(object sender, MouseButtonEventArgs e)

{

this.Close();

LogIn log = new LogIn();

Data.UserLogin = false;

Data.UserIsAdmin = false;

log.Show();

}

}

}

Листинг 4: Класс DataManagerRouteByNum

using System;

using System.Collections.Generic;

using System.Text;

using System.Collections.ObjectModel;

using AllTrans.Model;

using System.ComponentModel;

using System.Linq;

namespace AllTrans.ViewModel

{

public class DataManagerRouteByNumber: INotifyPropertyChanged

{

public event PropertyChangedEventHandler? PropertyChanged;

private void NotifyPropertyChange(string propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

public void StopPick(string routename)

{

\_stops = new ObservableCollection<Stop>(DataWorker.GetStopsByRoute(routename));

RefreshStops(routename);

}

public void RoutePick(int id)

{

\_transRoutesNames = new ObservableCollection<TransRoute>(DataWorker.GetRoutesById(id));

RefreshRoute(id);

}

private ObservableCollection<TransRoute> \_transRoutesNames = new ObservableCollection<TransRoute>(DataWorker.GetRoutesById(0));

public ObservableCollection<TransRoute> TransRoutes { get { return \_transRoutesNames; } }

private ObservableCollection<Stop> \_stops = new ObservableCollection<Stop>(DataWorker.GetStopsByRoute(""));

public ObservableCollection<Stop> Stops { get { return \_stops; } }

private ObservableCollection<StopTime> \_time = new ObservableCollection<StopTime>(DataWorker.GetTimeByRouteAndStop(0,0));

public ObservableCollection<StopTime> Time { get { return \_time; } }

private ObservableCollection<Transport> \_transport = new ObservableCollection<Transport>(DataWorker.GetNumbersListByRoute(""));

public ObservableCollection<Transport> Transports { get { return \_transport; } }

public void RefreshStops(string routename)

{

\_stops = new ObservableCollection<Stop>(DataWorker.GetStopsByRoute(routename));

NotifyPropertyChange("Stops");

}

public void RefreshRoute(int id)

{

\_transRoutesNames = new ObservableCollection<TransRoute>(DataWorker.GetRoutesById(id));

NotifyPropertyChange("TransRoutes");

}

public void RefreshTime(int id, int sid)

{

\_time = new ObservableCollection<StopTime>(DataWorker.GetTimeByRouteAndStop(id, sid));

NotifyPropertyChange("Time");

}

public void SetTime(int id)

{

using (ApplicationContext db = new ApplicationContext())

{

var time = db.StopsTime.Where(el => el.id == id);

}

NotifyPropertyChange("Time");

}

}

}

Листинг 5: Класс ReserveTicketForm

using System;

using System.Collections.Generic;

using System.Text;

using System.Collections.ObjectModel;

using AllTrans.Model;

using System.ComponentModel;

namespace AllTrans.ViewModel

{

public class DataManagerStopsPick: INotifyPropertyChanged

{

public event PropertyChangedEventHandler? PropertyChanged;

private void NotifyPropertyChange(string propertyName)

{

if (PropertyChanged != null)

{

PropertyChanged(this, new PropertyChangedEventArgs(propertyName));

}

}

public void RoutePick(string routename)

{

\_stops = new ObservableCollection<Stop>(DataWorker.GetStopsByRoute(routename));

RefreshStops(routename);

}

public void NumberPick(string routename)

{

\_transport = new ObservableCollection<Transport>(DataWorker.GetNumbersListByRoute(routename));

RefreshTransport(routename);

}

private ObservableCollection<TransRoute> \_transRoutesNames = new ObservableCollection<TransRoute>(DataWorker.GetRoutesWONum());

public ObservableCollection<TransRoute> TransRoutes { get { return \_transRoutesNames; } }

private ObservableCollection<Stop> \_stops = new ObservableCollection<Stop>(DataWorker.GetStopsByRoute(""));

public ObservableCollection<Stop> Stops { get { return \_stops; } }

private ObservableCollection<Transport> \_transport = new ObservableCollection<Transport>(DataWorker.GetNumbersListByRoute(""));

public ObservableCollection<Transport> Transports { get { return \_transport; } }

public void RefreshStops(string routename)

{

\_stops = new ObservableCollection<Stop>(DataWorker.GetStopsByRoute(routename));

NotifyPropertyChange("Stops");

}

public void RefreshTransport(string routename)

{

\_transport = new ObservableCollection<Transport>(DataWorker.GetNumbersListByRoute(routename));

NotifyPropertyChange("Transports");

}

}

}

Листинг 6: Класс DataManagerNumberPick

using System;

using System.Collections.Generic;

using System.Text;

using System.Collections.ObjectModel;

using AllTrans.Model;

namespace AllTrans.ViewModel

{

class DataManagerNumberPick

{

private ObservableCollection<Transport> \_transport = DataWorker.GetNumbersList(Data.SelectedType);

public ObservableCollection<Transport> Transports{ get { return \_transport; } }

}

}

Листинг 7: Класс StopsPickPage

using AllTrans.Model;

using AllTrans.ViewModel;

using System.Windows;

using System.Windows.Controls;

namespace AllTrans.View

{

/// <summary>

/// Логика взаимодействия для StopsPickPage.xaml

/// </summary>

public partial class StopsPickPage : Page

{

DataManagerStopsPick stopsPick = new DataManagerStopsPick();

public StopsPickPage()

{

InitializeComponent();

this.DataContext = stopsPick;

}

private void SelectionRouteEvent(object sender, SelectionChangedEventArgs e)

{

stopsPick.RoutePick(((TransRoute)this.RouteList.SelectedItem).name);

stopsPick.NumberPick(((TransRoute)this.RouteList.SelectedItem).name);

}

private void PickEvent(object sender, System.Windows.Input.MouseButtonEventArgs e)

{

if(this.RouteList.SelectedIndex%2 ==0)

{

NavigationService.Navigate(new RoutePickPage(((Transport)this.TimeList.SelectedItem).id, 0, this.StopsList.SelectedIndex));

}

else

{

NavigationService.Navigate(new RoutePickPage(((Transport)this.TimeList.SelectedItem).id, 1, this.StopsList.SelectedIndex));

}

}

}

}