МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Филиал федерального государственного бюджетного образовательного учреждения высшего образования

«Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

Направление: 09.03.01 «Информатика и вычислительная техника»

Курсовая работа

по предмету: «Технология программирования»

Тема**:** «Автоматизированная информационная система продажи

авиабилетов**»**

Студент \_\_\_\_ИВТ1-19\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_Милославский С. А.\_

группа подпись фамилия И.О.

Руководитель \_\_к. т. н.\_\_\_\_\_ \_\_доцент\_\_ \_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_Федулов Я. А.\_\_\_

учен. степень должность подпись фамилия И.О.

Смоленск – 2021

АННОТАЦИЯ

Пояснительная записка содержит 150 листов печатного текста, 24 рисунков, 13 таблицу, список литературы из 20 наименований и 3 формулы.

Данный курсовой проект состоит из введения, трёх разделов: анализ технического задания, проектирование разрабатываемой системы и реализация разрабатываемой системы; заключения, списка используемых источников и приложения.

В первом разделе анализируется техническое здание курсового проекта и объясняется выбор средств разработки проекта.

Во втором разделе разбирается предметная область курсового проекта и для наглядного представления бизнес логики разрабатываемой системы строятся следующие виды диаграмм:

* функциональная диаграмма IDEF0;
* диаграмма переходов состояний STD;
* диаграмма вариантов использования;
* диаграмма деятельности;
* диаграмма последовательности действий;
* диаграмма Джексона.

Кроме этого во втором разделе приводятся схемы алгоритмов, отражающие выполнение и работу проекта.

В третьем разделе осуществляется сама реализация разрабатываемой системы. Строятся две диаграммы: классов и компонентов, кроме этого в этом разделе приводятся основные интерфейсные формы проекта, строится граф диалога и выполняется тестирование проекта двумя способами: структурное и функциональное тестирование.

Структурное тестирование включает в себя тестирование:

* базового пути;
* условий;
* циклов.

Функционально тестирование включает в себя:

* разбиение на классы эквивалентности;
* анализ граничных значений;
* анализ причинно-следственных связей.

Кроме этого в третьем разделе рассчитывается оценка качества программных средств – ГОСТ 28195-89.

Объект исследования: программные средства для продаж авиабилетов.

Цель работы: разработать автоматизированную информационную систему продажи авиабилетов.

Технология разработки – Visual Studio 2019, PostgreSQL server 13.0, C#

Результаты работы: спроектирована и разработана автоматизированная информационная система продажи авиабилетов.

ANNOTATION

The explanatory note contains 150 sheets of printed text, 24 pictures, 13 tables, a bibliography of 20 titles and 3 formulas.

This course project consists of an introduction, three sections: analysis of technical specifications, design of the system under development and implementation of the system under development; conclusion, list of sources used and applications.

The first section analyzes the technical building of the course design and explains the choice of project development tools.

I In the second section, the subject area of the course project is analyzed and the following types of diagrams are built to visualize the business logic of the system being developed:

* functional diagram IDEF0;
* STD state transition diagram;
* use case diagram;
* activity diagram;
* sequence diagram;
* Jackson diagram.

In addition, the second section provides diagrams of algorithms that reflect the implementation and operation of the project.

In the third section, the implementation of the system being developed is carried out. Two diagrams are built: classes and components, in addition, in this section, the main interface forms of the project are given, a dialog graph is built and the project is tested in two ways: structural and functional testing.

Structural testing includes testing:

* base path;
* conditions;
* cycles.

Functional testing includes:

* partition into equivalence classes;
* analysis of boundary values;
* analysis of cause-and-effect relationships.

In addition, in the third section, an assessment of the quality of software is calculated - GOST 28195-89.

Research object: software for air ticket sales.

Purpose of work: to develop an automated information system for the sale of air tickets.

Development technology - Visual Studio 2019, PostgreSQL server 13.0, C #

Results of work: designed and developed an automated information system for the sale of air tickets.

Содержание

[ВВЕДЕНИЕ 7](#_Toc74218913)

[1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ 9](#_Toc74218914)

[1.1 Выбор средств разработки 10](#_Toc74218915)

[1.2 Выводы по разделу 12](#_Toc74218916)

[2 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ 13](#_Toc74218917)

[2.1 Анализ предметной области 13](#_Toc74218918)

[2.2 Функциональная диаграмма IDEF0 14](#_Toc74218919)

[2.3 Диаграмма переходов состояний 21](#_Toc74218920)

[2.4 Функциональные возможности разрабатываемой системы 22](#_Toc74218921)

[2.5 Динамические аспекты поведения разрабатываемой системы. 25](#_Toc74218922)

[2.6 Интерактивное поведение разрабатываемой системы 28](#_Toc74218923)

[2.7 Диаграмма Джексона 30](#_Toc74218924)

[2.8 Схемы алгоритмов 31](#_Toc74218925)

[2.9 Выводы по разделу 33](#_Toc74218926)

[3 РЕАЛАЛИЗАЦИЯ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ 34](#_Toc74218927)

[3.1 Структурные или функциональные схемы 34](#_Toc74218928)

[3.1.1 Диаграмма компонентов 34](#_Toc74218929)

[3.1.2 Диаграмма классов 35](#_Toc74218930)

[3.2 Реализация пользовательского интерфейса 36](#_Toc74218931)

[3.3 Тестирование 37](#_Toc74218932)

[3.3.1 Структурное тестирование 37](#_Toc74218933)

[3.3.1.1 Тестирование базового пути 38](#_Toc74218934)

[3.3.1.2 Тестирование условий 39](#_Toc74218935)

[3.3.1.3 Тестирование циклов 40](#_Toc74218936)

[3.3.2 Функциональное тестирование 40](#_Toc74218937)

[3.3.2.1 Разбиение на классы эквивалентности и анализ граничных значений 41](#_Toc74218938)

[3.4 Оценка качества разрабатываемой системы 42](#_Toc74218939)

[3.5 Выводы по разделу 60](#_Toc74218940)

[ЗАКЛЮЧЕНИЕ 62](#_Toc74218941)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 63](#_Toc74218942)

[ПРИЛОЖЕНИЕ А – КОД ПРОГРАММЫ 65](#_Toc74218943)

[ПРИЛОЖЕНИЕ Б – ДИСК С разработанной системой 149](#_Toc74218944)

ВВЕДЕНИЕ

В настоящее время информационные системы занимают значимое место в жизни людей. Самые первые из них были созданы еще в 50-х годах прошлого столетия и осуществляли в основном арифметические расчеты, незначительно сокращая издержки производства и затраты времени. Развитие информационных систем не стояло на месте, продвигаясь в ногу со временем и деловыми потребностями человека. К банальным возможностям расчета зарплат добавились возможности анализировать информацию, упрощая процесс принятия решений для управленческого персонала. Также, с каждым годом степень автоматизации систем увеличивалась, позволяя все сильнее наращивать производственные показатели предприятий их использующих.

В современных условиях, человек вынужден работать с гигантскими объемами информации. В связи с этим разработка программных продуктов, служащих для автоматизированного учета, весьма актуальна. Системы обязаны представлять собой мощные средства, способные обрабатывать гигантские потоки данных высокой структурной сложности за минимум затраченного времени, обеспечивая дружественный диалог с пользователем.

На данный момент существует огромное количество информационных систем, занимающихся продажами авиабилетов. Они могут представлять собой как самостоятельные приложения, так и онлайн сервисы, предоставляя пользователю доступ к веб-службам поставщиков.

Целью данной курсовой работы является создание автоматизированной информационной системы, осуществляющей продажи авиабилетов.

Разработка подобной системы весьма актуальна на данный момент. В современном мире самолеты являются не только самым быстрым видом транспорта, но и самым безопасным, в связи с этим авиаперелеты пользуются весьма высокой популярностью. Вследствие этого продаваемые на рейсы билеты востребованы и с высокой вероятностью найдут своего покупателя, при условии, что авиакомпания обеспечила клиенту полноценный доступ к нужной ему информации. Это и есть задача, решаемая современными автоматизированными информационными системами. Существует множество подобных разработок, позволяющих авиакомпаниям реализовывать авиабилеты, а пользователям приобретать их. Однако, зачастую, функциональность таких систем либо весьма ограничена, либо предоставляет достаточное количество информации, жертвуя дружественностью к пользователю.

Для реализации поставленной задачи были выбраны следующие средства разработки: MS Visual Studio 2019, PostgreSQL server 13.0 и язык программирования C#.

1 АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Разрабатываемая система должна содержать в себе следующие подсистемы:

* подсистема администрирования, позволяющая осуществлять настройку системы и ее поддержку;
* клиентская подсистема, позволяющая просматривать справочную информацию и отправлять запросы на бронирование или возврат авиабилетов.

Для доступа к любой из данных подсистем пользователь должен пройти предварительную регистрацию или, если пользователь уже зарегистрирован, авторизацию. Разным группам пользователей доступны разные функциональные возможности и уровень доступа к информации.

В ходе данной курсовой работы должна быть создана автоматизированная информационная система продажи авиабилетов, решающая следующие задачи:

* продажа авиабилетов на запланированные рейсы;
* поиск авиабилетов по запросу пользователя;
* поиск подходящего рейса при поступлении заявки;
* администрирование информационной системы;
* создание приложения, предоставляющего пользователям графический интерфейс для доступа к системе.

Архитектурой информационной системы продажи авиабилетов называется концепция, определяющая структуру и взаимосвязь компонентов в данной системе. Компоненты информационной системы по выполняемым функциям можно разделить на три слоя: слой представления, слой бизнес-логики и слой доступа к данным. Разрабатываемой информационной системе продажи авиабилетов соответствует клиент-серверная архитектура (рисунок 1). [5]



Рисунок 1 – Архитектура клиент-сервер

Такой подход обеспечивает многопользовательский режим доступа к информации, а также гарантирует целостность данных. Пользователь использует спроектированный графический интерфейс клиентской части системы, которая в свою очередь отправляет запросы серверному ПО. Также особенностью такой архитектуры является разделение функциональных возможностей приложения между клиентом и сервером.

1.1 Выбор средств разработки

Согласно техническому заданию необходимо разработать автоматизированная информационная система продажи авиабилетов. Для реализации такой системы были выбраны следующие средства для разработки: MS Visual Studio 2019, PostgreSQL server 13.0 и язык программирования C#.

Microsoft Visual Studio – это интегрированная среда разработки (IDE) от Microsoft. Он используется для разработки компьютерных программ, а также веб-сайтов, веб-приложений, веб-сервисов и мобильных приложений. Visual Studio включает редактор кода, поддерживающий IntelliSense (компонент завершения кода), а также рефакторинг кода. Интегрированный отладчик работает как отладчик на уровне исходного кода, так и как отладчик на уровне машины. Другие встроенные инструменты включают профилировщик кода, конструктор для создания приложений с графическим интерфейсом, веб-дизайнер, конструктор классов и конструктор схемы базы данных. Данная среда разработки используется из-за ее возможностей в области работы с пользовательским интерфейсом, рефакторинга кода и удобной отладки.

Microsoft Visio – это приложение для создания диаграмм и векторной графики, входящее в семейство Microsoft Office. Для разработки схем процессов не требуется специальное обучение. Построение диаграмм и схем процессов осуществляется с помощью простого и понятного интерфейса. Одним из основных преимуществ Microsoft Visio является большое количество различных шаблонов диаграмм, что упрощает и ускоряет процесс создания схем бизнес процессов (например, eEPC, IDEF0, IDEF3, UML). Данное приложение использовалось для построения диаграмм и схем в данном проекте.

PostgreSQL (Postgres) – система управления реляционными базами данных (RDBMS) с открытым исходным кодом, в которой особое внимание уделяется расширяемости и совместимости с SQL. Эта система предназначена для обработки и хранения данных. В данном проекте служит хранилищем данных. [3]

pgAdmin – инструмент для администрирования данных на сервере PostgreSQL. Он представляет собой графический клиент для работы с сервером, через который можно с легкостью создавать, удалять, изменять базы данных и управлять данными. В проекте этот инструмент использовался для проверки и манипуляции данными.

Redis – хранилище структуры данных, используемое в качестве распределенной базы данных ключ-значение, кеша и брокера сообщений с дополнительной надежностью. В проекте используется для аутенфикации. [8]

Windows Subsystem for Linux (WSL) – это уровень совместимости для запуска Linux приложений в ОС Windows 10. В проекте используется для удобной работы с Redis на Windows.

GraphQL – это язык запросов и обработки данных с открытым исходным кодом для API, а также среда выполнения для выполнения запросов с существующими данными.

1.2 Выводы по разделу

В этом разделе курсового проекта анализировалось техническое задание. Из приведенного анализа был спроектирован необходимый функционал для автоматизированной информационной системы продажи авиабилетов.

Кроме этого в этом разделе описан выбор инструментальных средств и разобраны основные средства компоненты, которые участвуют в реализации проекта.

2 ПРОЕКТИРОВАНИЕ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ

2.1 Анализ предметной области

Целью данной курсовой работы является создание программного средства, осуществляющего поиск и продажу авиабилетов.

Авиакомпания занимается авиаперевозками пассажиров. Также она устанавливает маршруты полетов. Рейсы осуществляются по установленным маршрутам согласно расписанию. На каждый рейс существует определенное количество билетов. Продажа билета пользователю осуществляется при отправке запроса на бронирование, при условии, что данный билет до сих пор есть в наличии. Приобретя билет, пользователь предоставляет информацию о себе и становится пассажиром. Совершеннолетние пассажиры обязаны иметь занесенные в базу данных (БД) паспортные данные. Несовершеннолетние обязаны иметь занесенные в БД данные из свидетельства о рождении. Администраторы системы могут ограничивать или расширять доступ пользователей и сотрудников к предоставляемой информации.

Система создаётся для обслуживания следующих групп пользователей:

* пользователи, приобретающие билеты на рейсы и осуществляющие их поиск;
* администраторы, осуществляющие контроль за пользователями и функциональностью системы.

Абсолютно каждая авиакомпания использует определенную систему дистрибуции. Наиболее развитые используют GDS (глобальные дистрибьюторские системы, которые формируются из основных международных компьютерных систем резервирования). В итоге сервисы продаж авиабилетов при поиске информации пользуются ресурсами глобальных дистрибьюторских систем. Однако доступ к GDS является не бесплатным, поэтому в роли дистрибутивной системы для разрабатываемой системы будет выступать БД, созданная в PostgreSQL server 13.0. [7]

В БД должна храниться информация:

* о маршрутах;
* о рейсах;
* о пользователях;
* о билетах;
* об авиакомпаниях;
* о сотрудниках;
* о пассажирах.

Разрабатываемая информационная система предназначена для продажи авиабилетов и упрощения доступа к нужной информации. Наличие данной разработки улучшает организационную работу авиаперевозчика за счёт отсутствия бумажной документации, поиск и систематизация которой занимали бы очень большое количество времени

2.2 Функциональная диаграмма IDEF0

IDEF0 – нотация графического моделирования, используемая для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающих эти функции. Тем не менее методология IDEF0 является одним из популярных подходов для описания бизнес-процессов. К ее особенностям можно отнести:

* использование контекстной диаграммы;
* поддержка декомпозиции;
* доминирование;
* выделение четырех типов стрелок.

Контекстная диаграмма – самая верхняя диаграмма, на которой объект моделирования представлен единственным блоком с граничными стрелками (в каждой модели IDEF0 может быть только одна контекстная диаграмма). Эта диаграмма называется A-0 (А минус нуль). Стрелки на этой диаграмме отображают связи объекта моделирования с окружающей средой. Диаграмма A-0 устанавливает область моделирования и ее границу. [9]

Нотация IDEF0 поддерживает последовательную декомпозицию процесса до требуемого уровня детализации. Дочерняя диаграмма, создаваемая при декомпозиции, охватывает ту же область, что и родительский процесс, но описывает ее более подробно. Согласно методологии, IDEF0 при декомпозиции стрелки родительского процесса переносятся на дочернюю диаграмму в виде граничных стрелок.

В данном пункте пояснительной записки приводится диаграмма IDEF0, которая разделена на три уровня иерархии диаграммы.

На рисунке 2.2.1 представлена контекстная диаграмма «Автоматизированная информационная система продажи авиабилетов», отражающая нулевой уровень иерархии. Из этой диаграммы видно, как разрабатываемая система взаимодействует с данными, и кто может к ней иметь доступ.

На рисунке 2.2.2 представлена диаграмма декомпозиции «Автоматизированная информационная система продажи авиабилетов», которая отражает второй уровень иерархии. Эта диаграмма представляет собой подробное описание контекстной диаграммы (рисунок 2.2.1). Из этой диаграммы видно, как разрабатываемая система взаимодействует с пользователями и какой доступ к данным они имеют.

Аналогичным образом строились диаграммы декомпозиции, представленные на рисунках 2.2.3–2.2.5, отражающие третий уровень иерархии. Из этих диаграмм видно, как данные взаимодействуют с БД и какие права имеют пользователи этой системы.



Рисунок 2.2.1 – Контекстная диаграмма «Автоматизированная информационная система продажи авиабилетов»



Рисунок 2.2.2 – Диаграмма декомпозиция модуля «Автоматизированная информационная система продажи авиабилетов»



Рисунок 2.2.3 – Диаграмма декомпозиция модуля «Аутенфикация в системе»



Рисунок 2.2.4 – Диаграмма декомпозиция модуля «Администрирование системы»



Рисунок 2.2.5 – Диаграмма декомпозиция модуля «Покупка авиабилета»

2.3 Диаграмма переходов состояний

Диаграммы переходов состояний (STD) предназначены для моделирования и документирования аспектов систем, зависящих от времени или реакции на событие. Они позволяют осуществлять декомпозицию управляющих процессов и описывают отношения между входными и выходными управляющими потоками для управляющего процесса-предка. [16]

С помощью STD-диаграмм можно моделировать последующее функционирование системы на основе ее предыдущего и текущего функционирования. Моделируемая система в любой заданный момент времени находится точно в одном из конечного множества состояний. С течением времени она может изменить свое состояние, при этом переходы между состояниями должны быть точно определены. На рисунке 2.3 представлена диаграмма состояний для окна «Покупка авиабилета», отражающая состояние перехода процессов для интерфейсной формы «Покупка авиабилета».



Рисунок 2.3 – Диаграмма состояний для окна «Покупка авиабилета»

2.4 Функциональные возможности разрабатываемой системы

Функциональные возможности системы представлены в диаграмме вариантов использования (рисунок 2.4).

Диаграмма вариантов использования показывает отношения между актерами и прецедентами, описывая систему на концептуальном уровне. В данном случае существует девять актеров:

* пользователь;
* зарегистрированный пользователь;
* незарегистрированный пользователь;
* незарегистрированный сотрудник;
* клиент;
* кассир;
* диспетчер;
* работник отдела кадров;
* системный администратор.

Причем для предотвращения дублирования информации было использовано отношение обобщения.

Представленные на диаграмме варианты использования показывают с какими данными могут взаимодействовать актеры и какие действия они могут выполнять над этими данными. Кроме того, что варианты использования показывают действия актеров над данными, так они отражают своего рода права доступа актеров к предоставленным им данным в разрабатываемой системе. [12]

Для каждого варианта использования должна строиться таблица, отражающая в себе следующую информацию:

* название;
* действующие лица
* цель;
* сценарий;
* результат.



Рисунок 2.4 – Диаграмма вариантов использования

В данном пункте пояснительной записки приведена часть таких таблиц.

В таблицах 2.4.1 и 2.4.2 описаны варианты использования «Регистрация» и «Войти в систему», представленные на рисунке 2.4.1.

Таблица 2.4.1 – Краткое описание варианта использования «Регистрация»

|  |  |
| --- | --- |
| Название | Регистрация |
| Действующие лица | Пользователь, «АТО» |
| Цель | Создать нового пользователя «АТО» |
| Предусловие | Нет |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь нажимает на кнопку «Создать аккаунт». 3. Программа открывает новое окно регистрации. Окно входа в систему скрывается. 4. Пользователь заполняет все необходимые поля: Логин, Пароль, Имя, Фамилия, Отчество. 5. Пользователь нажимает на кнопку «Регистрация». 6. Программа проверяет правильность ввода, и в случае успешной проверки данные пользователя добавляются в соответствующие таблицы БД. 7. Окно регистрации закрывается, и вновь отрывается окно входа в программу. |
| Результат | Пользователь добавлен в систему. |

Таблица 2.4.2 – Краткое описание варианта использования «Войти в систему»

|  |  |
| --- | --- |
| Название | Войти в систему |
| Действующие лица | Зарегистрированный пользователь, «АТО» |
| Цель | Войти в систему «АТО» |
| Предусловие | Пользователь ранее был зарегистрирован в системе «АТО» |
| Успешный сценарий | 1. Пользователь запускает программу и видит окно входа в аккаунт. 2. Пользователь вводит свой логин и пароль. 3. Пользователь нажимает на кнопку «Войти». 4. Программа проверяет наличие пользователя с таким логином и паролем, и в случае успешной проверки пользователь входит в свой аккаунт. 5. Окно входа в аккаунт срывается. 6. Отрывается окно с дополнительными возможностями (это окно зависит от типа пользователя). |
| Результат | Пользователь вошел в свой аккаунт «АТО» и получил дополнительные возможности в зависимости от прав доступа.. |

Построенные таблицы вариантов использования отражают подробное описание определенного варианта использования, изображенного на рисунке 2.4.1. Аналогичным образом строятся таблицы для остальных вариантов использования.

2.5 Динамические аспекты поведения разрабатываемой системы.

Динамические аспекты поведения системы представлены в диаграмме активности.

Диаграмма активности позволяет построить графическое представление потока от одного действия к другому. Действие можно описать как работу системы. Поток управления перенаправляется от одной операции к другой. Этот поток может быть:

* последовательным;
* разветвленным;
* параллельным.

Диаграммы активности в основном используются в качестве блок-схемы, которая состоит из действий, выполняемых системой. Она фиксирует динамическое поведение системы, иными словами используется для отображения потока сообщений от одного действия к другому. [15]

Активности на диаграмме «разбросаны» по беговым дорожкам, каждая из которых соответствует поведению одного из объектов. Благодаря этому легко определить, каким из объектов выполняется каждая из активностей.

Дорожка – часть области диаграммы деятельности, на которой отображаются только те активности, за которые отвечает конкретный объект. Предназначены дорожки для разбиения диаграммы в соответствии с распределением ответственности за действия, выполняемых разными участниками в одном потоке. Имя дорожки может означать роль или объект, которому она соответствует.

На рисунках 2.5.1 и 2.5.2 представлены диаграммы активности «Войти в систему» и «Заказ авиабилета». Эти диаграммы отражают пошаговое выполнение действий вариантов использования представленных на рисунке 2.4.1. Кроме пошагово выполнения действий на этих диаграммах присутствуют дорожки «Пользователь» и «АТО», которые разбивают диаграмму на группировки действий. Представленное на диаграммах разбиение действий на группировки показывает какой объект отвечает за какие процессы выполнения действий.



Рисунок 2.5.1 – Диаграмма активности «Войти в систему»



Рисунок 2.5.2 – Диаграмма активности «Заказ авиабилета»

2.6 Интерактивное поведение разрабатываемой системы

Интерактивное поведение системы представлены в диаграмме последовательности.

Диаграмма последовательности показывает взаимодействие между объектами в последовательном порядке, то есть в порядке, в котором эти взаимодействия происходят. На ней изображаются объекты, задействованные в сценарии, и последовательность сообщений, которыми обмениваются объекты, необходимые для выполнения функций сценария. Диаграммы последовательностей обычно связаны с реализациями вариантов использования в логическом представлении разрабатываемой системы. [2]

Диаграмма последовательности показывает в виде параллельных вертикальных линий (линий жизни) разные процессы или объекты, которые существуют одновременно, а в виде горизонтальных стрелок – сообщения, которыми они обмениваются, в том порядке, в котором они происходят. Это позволяет описывать простые сценарии выполнения в графическом виде.

На рисунках 2.6.1 и 2.6.2 представлены диаграммы последовательности «Авторизация в системе» и «Бронирование авиабилета». Эти диаграммы отражают взаимодействие пользователя с разрабатываемой системой. Из-за выбранной архитектуры клиент-сервер принцип работы на клиенте одинаков, в связи с этим диаграммы последовательности похожи друг на друга, кроме формы взаимодействия с пользователем.

По представленным диаграммам последовательности видно, как осуществляется выполнение запросов на клиенте, а именно как данные взаимодействует через изображенные объекты. Выполнение запросов происходит следующем образом.

Пользователь взаимодействует с формой, то есть заполняет форму необходимыми данными. Далее выполняется соответствующий запрос, который передает введенные пользователем данные на форме серверу. Сервер обрабатывает полученные данные и тоже вызывает соответствующий запрос, который обращается уже к БД. После обработки запроса в БД пользователь получает ответ на свои действия.



Рисунок 2.6.1 – Диаграмма последовательности «Авторизация в системе»



Рисунок 2.6.2 – Диаграмма последовательности «Бронирование авиабилета»

2.7 Диаграмма Джексона

Структурированное программирование Джексона (JSP) – это метод структурного программирования, разработанный британским консультантом по программному обеспечению Майклом А. Джексоном и описанный в его книге. Техника JSP заключается в анализе структур данных файлов, которые программа должна читать как ввод и создавать как вывод, а затем создавать проект программы на основе этих структур данных, чтобы структура управления программой обрабатывала эти структуры данных. естественным и интуитивно понятным способом. JSP описывает структуры (как данных, так и программ) с использованием трех основных структур:

* последовательность;
* итерация;
* выбор (или альтернативы).

Эти структуры представлены в виде (по сути) визуального представления регулярного выражения. [6]

На рисунке 2.7 представлена диаграмма Джексона «Покупка авиабилета».



Рисунок 2.7 – Диаграмма Джексона

2.8 Схемы алгоритмов

Написание логического пошагового метода решения проблемы называется алгоритмом. Другими словами, алгоритм – это процедура решения проблем. Это первый шаг в решении математической или компьютерной задачи. [11]

Алгоритм включает вычисления, рассуждения и обработку данных. Алгоритмы могут быть представлены естественными языками, псевдокодом, блок-схемами и т. д.

Схема алгоритма – это графическое представление алгоритма с помощью различных символов, форм и стрелок для демонстрации процесса или программы. С помощью алгоритмов можно легко понять программу. Основная цель использования схемы алгоритма – анализ различных методов.

В данном пункте представлены два алгоритма «Авторизация в системе» и «Бронирование авиабилета».

Алгоритм «Авторизация в системе» применяется для входа в систему «АТО». Он заключается в проверке существования данных пользователя в БД разрабатываемой системы. Схема этого алгоритма представлена на рисунке 2.8.1.



Рисунок 2.8.1 – Схема алгоритма «Авторизация в системе»

На рисунке 2.8.1 проверка существования пользователя представляет собой запрос к необходимым данным пользователя, которые хранятся в БД. Если запрос выполнен успешно, то пользователю предоставляется доступ в систему, иначе система отобразить соответствующее сообщение об ошибке.

Алгоритм «Бронирование авиабилета» применяется для покупки авиабилетов. Он заключается в проверке правильности заполнения необходимых полей и поиском подходящего рейса. Схема этого алгоритма представлена на рисунке 2.8.2.



Рисунок 2.8.2 – Схема алгоритма «Бронирование авиабилета»

На рисунке 2.8.2 поиск рейса представляет собой запрос, который обращается к БД и подбирает ближайший рейс.

2.9 Выводы по разделу

В этом разделе курсового проекта осуществлялось проектирование разрабатываемой системы «АТО». Для наглядного представления и определения необходимых требований к разработанной системе были построение следующие диаграммы:

* диаграмма IDEF0;
* диаграмма переходов состояний;
* диаграмма вариантов использования;
* диаграмма активности;
* диаграмма последовательности;
* диаграмма Джексона.

Кроме этого в этом разделе разбирались и приводились использованные алгоритмы в разрабатываемой системе.

3 РЕАЛАЛИЗАЦИЯ РАЗРАБАТЫВАЕМОЙ СИСТЕМЫ

3.1 Структурные или функциональные схемы

3.1.1 Диаграмма компонентов

Диаграмма компонентов описывает организацию и подключение физических компонентов в системе, то есть показывает структурные отношения между компонентами системы. Диаграммы компонентов используется, чтобы помочь в деталях реализации модели и перепроверить, что каждый аспект требуемых функций системы охвачен запланированной разработкой. [13]

В этой диаграмме компоненты являются менее физическими и более концептуальными автономными элементами дизайна, такими как бизнес-процесс, который предоставляет или требует интерфейсов для взаимодействия с другими конструкциями в системе. Физические элементы, такие как файлы и документы, теперь называются артефактами.

На рисунке 3.1.1.1 представлена диаграмма компонентов «Клиент-сервер». Эта диаграмма отображает взаимосвязь между клиентской и серверной частью.



Рисунок 3.1.1.1 – Диаграмма компонентов «Клиент-сервер»

На рисунках 3.1.1.2 и 3.1.1.3 представлены подробные диаграммы компонентов клиентской и серверной части. Диаграммы серверной и клиентвской части отображают, как компоненты взаимодействуют между собой и в какой последовательности выполняются. По этим диаграмма видно, как клиентская часть предоставляет интерфейс серверной части.



Рисунок 3.1.1.2 – Диаграмма компонентов серверной части системы



Рисунок 3.1.1.3 – Диаграмма компонентов клиентской части системы

3.1.2 Диаграмма классов

Диаграмма классов – статическая структура, которая дает обзор программного обеспечения системы, отображая классы, атрибуты, операции и их отношения между собой. Эта диаграмма включает имя класса, атрибуты и операцию в отдельных обозначенных отсеках. Диаграмма классов помогает построить код для разработки программного обеспечения.

Диаграмма классов определяет типы объектов в системе и различные типы отношений, которые существуют между ними. Это дает общее представление о приложении. Этот метод моделирования может работать практически со всеми объектно-ориентированными методами. Класс может относиться к другому классу. Класс может иметь свои объекты или может наследовать от других классов. [4]

На рисунке 3.1.2 представлена диаграмма классов. Эта диаграмма отображает классы моделей сервера с их атрибутами.

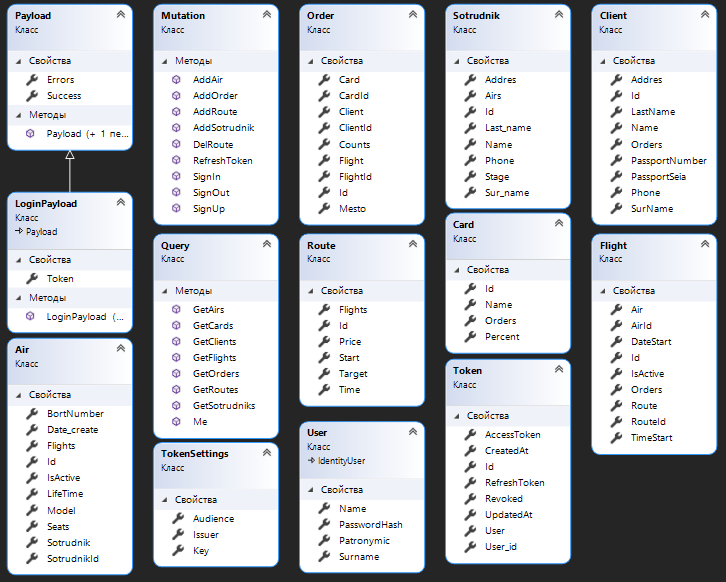


Рисунок 3.1.2 – Диаграмма классов «Сервер»

3.2 Реализация пользовательского интерфейса

Граф диалога – ориентированный взвешенный граф, каждой вершине которого соответствует определенное состояние диалога, характеризующееся набором доступных пользователю действий. Дуги, исходящие из вершин, показывают возможные изменения состояний при выполнении пользователем указанных действий. Таким образом, граф представляет собой набор состояний системы, между которыми в ходе диалога при определенных условиях осуществляются переходы.

Разработка графа диалога позволяет выявить и устранить возможные тупиковые ситуации, выбрать рациональный путь перехода из текущего состояния системы в требуемое, выявить неоднозначные ситуации, когда для пользователя требуется дополнительная помощь.

Интерфейс пользователя можно упростить, снизив степень неопределенности действий пользователя. Для этого можно применить смешанную структуру диалога, ограничив при необходимости свободу выбора пользователя, используя меню или другие элементы и контролировать вводимую пользователем информацию, принимать только допустимые данные.

На рисунке 3.2 изображён граф диалога для формы покупки авиабилета. Этот граф отображает путь перехода из текущего состояния в требуемое для этой формы.



Рисунок 3.2 – Граф диалога формы покупки авиабилета

3.3 Тестирование

3.3.1 Структурное тестирование

Структурное тестирование – это тип тестирования, проводимого для проверки структуры кода. Это также известно, как тестирование «белого ящика» или «тестирование стеклянного ящика». Этот тип тестирования требует знания кода, поэтому в основном его проводят разработчики. Его больше заботит то, как это работает система, а не ее функциональные возможности. Это обеспечивает больший охват тестирования. [17]

Структурные методы тестирования:

* тестирование базового пути;
* тестирование условий;
* тестирование циклов.

Для этого метода тестирования был выбран алгоритм, изображенный на рисунке 2.8.1.

3.3.1.1 Тестирование базового пути

Тестирование пути – это метод структурного тестирования, который включает использование исходного кода программы для поиска всех возможных путей к исполняемому файлу. Это помогает определить все ошибки, лежащие в фрагменте кода. Этот метод предназначен для выполнения всего или выбранного пути через компьютерную программу.

Любая программа включает несколько точек входа и выхода. Проверка каждого из этих пунктов является сложной задачей и требует много времени. Чтобы уменьшить количество избыточных тестов и достичь максимального тестового покрытия, используется тестирование базового пути.



Рисунок 3.3.1.1.1 – Контрольный граф тестируемого алгоритма

Количество дуг E = 4; количество узлов N = 4. Далее выполняется расчет цикломатической сложности: V(G)=E-N+2 = 4 – 4 + 2 = 2.

После расчета цикломатической сложности находятся базовый набор путей. Независимые пути:

1. 1-2-3-4
2. 1-2-4

Таблица 3.3.1.1.1 – Результаты тестирования базового пути

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Цель | Ожидаемый результат | Полученный результат | Примечание |
| 1 | 1-2-3-4 | Доступ закрыт | Сообщение об ошибке (рисунок 3.3.1.1.2) | Корректный результат |
| 2 | 1-2-4 | Доступ открыт | Успешный вход в систему (рисунок 3.3.1.1.3) | Корректный результат |

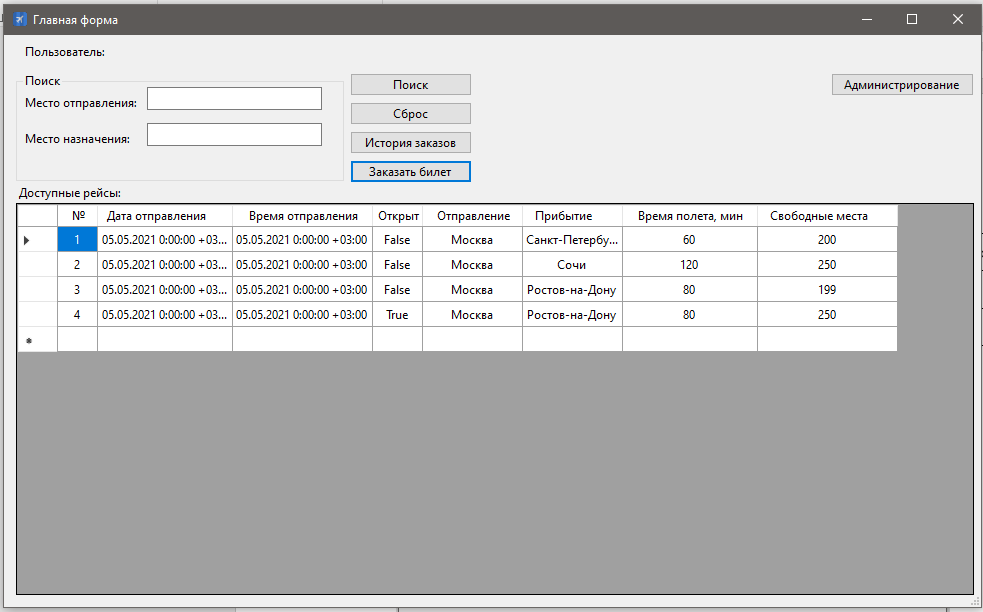


Рисунок 3.3.1.1.2 – Результат теста 1 (успешный вход в систему)

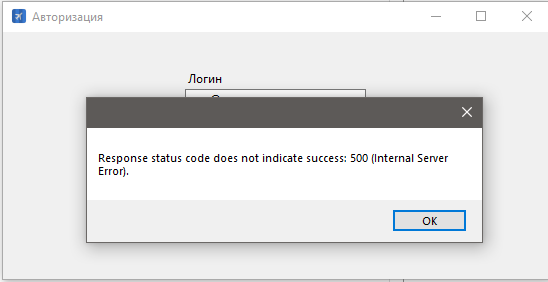


Рисунок 3.3.1.1.3 – Результат теста 2 (Сообщение об ошибке)

3.3.1.2 Тестирование условий

Условие тестирования – это особый набор ограничений, который может содержать такие функциональные возможности, как транзакции, функции или структурные элементы для тестирования программного приложения. Условия тестирования помогают убедиться, что в программном приложении отсутствуют ошибки.

В тестируемом алгоритме присутствует только одно условие и сценарий для этого тестирования простой. Есть ли доступ к системе у пользователя. Результаты представлены в таблице 3.3.1.2, где Е1 – вводимые данные, а Е2 – данные хранимые в БД.

Таблица 3.3.1.2 – Результаты тестирования условий

|  |  |  |  |
| --- | --- | --- | --- |
| № | E1 | E2 | E1 & E2 |
| 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 1 | 1 | 1 |

3.3.1.3 Тестирование циклов

Циклическое тестирование определяется как тип тестирования программного обеспечения, который полностью фокусируется на валидности конструкций цикла. В тестируемом алгоритме нет циклов.

3.3.2 Функциональное тестирование

Функциональное тестирование – это тип тестирования программного обеспечения, при котором проверяется соответствие системы программного обеспечения функциональным требованиям или спецификациям. Целью функциональных тестов является тестирование каждой функции программного приложения путем предоставления соответствующих входных данных и проверки выходных данных на соответствие функциональным требованиям. [1]

Функциональное тестирование в основном включает в себя тестирование черного ящика и не касается исходного кода приложения. Это тестирование проверяет пользовательский интерфейс, API, базу данных, безопасность, связь клиент или сервер и другие функции тестируемого приложения. Тестирование можно проводить вручную или с помощью автоматизации.

Для этого метода тестирования был выбран алгоритм, изображенный на рисунке 2.8.2.

3.3.2.1 Разбиение на классы эквивалентности и анализ граничных значений

Разделение классов эквивалентности – это тип метода тестирования черного ящика, который может применяться ко всем уровням тестирования программного обеспечения, таким как модульное тестирование, интеграция, система и т. д. В этом методе единицы входных данных делятся на эквивалентные разделы, которые могут использоваться для получения тестовые примеры, которые сокращают время, необходимое для тестирования, из-за небольшого количества тестовых примеров.

Граничное тестирование – это процесс тестирования между крайними точками или границами между разделами входных значений.

Тестовые варианты:

1. Бронирование двух авиабилета;
2. Бронирование одного авиабилетов.

Таблица 3.3.2.1 – Результаты функционального тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Цель | Ожидаемый результат | Полученный результат | Примечание |
| 1 | Забронировано 2 билета | Покупка оформлена | Результат тестов представлен на рисунке 3.3.2.1 | Корректный результат |
| 2 | Забронирован 1 билет | Покупка оформлена | Корректный результат |

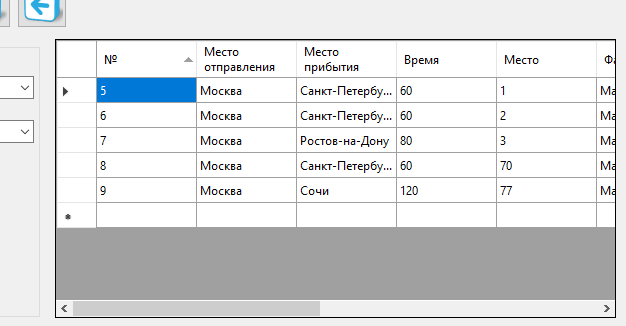


Рисунок 3.3.2.1 – Результат функционального тестирования

3.4 Оценка качества разрабатываемой системы

Оценка качества разрабатываемой системы выполняется согласно ГОСТ 28195-89. Этот стандарт, устанавливает общие положения по оценке качества программных средств вычислительной техники (ПС), поставляемых через фонды алгоритмов и программ (ФАП), номенклатуру и применяемость показателей качества ПС. [20]

Согласно госту 28195-89 был выбран 5012 (программные средства управления базами данных) подкласс показателей качества ПС. Показатели качества этого подкласса приведены в таблице 3.4.1.

Таблица 3.4.1 – Показателей качества программных средств подкласса 5012

|  |  |  |
| --- | --- | --- |
| Наименование групп и комплексных показателей качества | Обозначение показателя | Применяемость показателя по подклассам (группам) ПС |
| 1. Показатели надежности ПС | | |
| 1.1. Устойчивость функционирования | H1 | + |
| 1.2. Работоспособность | Н2 | + |
| 2. Показатели сопровождения | | |
| 2.1. Структурность | С1 | + - |
| 2.2. Простота конструкции | С2 | + - |

*Продолжение таблицы 3.4.1*

|  |  |  |
| --- | --- | --- |
| Наименование групп и комплексных показателей качества | Обозначение показателя | Применяемость показателя по подклассам (группам) ПС |
| 2.3. Наглядность | С3 | + - |
| 2.4. Повторяемость | С4 | + - |
| 3. Показатели удобства применения | | |
| 3.1. Легкость освоения | У1 | + - |
| 3.2. Доступность эксплуатационных программных документов | У2 | + |
| 3.3. Удобство эксплуатации и обслуживания | УЗ | + |
| 4. Показатели эффективности | | |
| 4.1. Уровень автоматизации | Э1 | + - |
| 4.2. Временная эффективность | Э2 | + - |
| 4.3. Ресурсоемкость | Э3 | + |
| 5. Показатели универсальности | | |
| 5.1. Гибкость | Г1 | + - |
| 5.2. Мобильность | Г2 | + - |
| 5.3. Модифицируемость | Г3 | + |
| 6. Показатели корректности | | |
| 6.1. Полнота реализации | К1 | + |
| 6.2. Согласованность | К2 | + |
| 6.3. Логическая корректность | К3 | + |
| 6.4. Проверенность | К4 | + |

В данных таблицах введены следующие обозначения:

 – весовой коэффициент для j-х критериев i-го фактора;

 – базовое значение для j-х критериев i-го фактора;

 – абсолютные показатели критериев для i-х факторов;

 – весовой коэффициент для j-х метрик i-го критерия;

 - значение, вычисленное из следующих таблиц по формуле для k-х метрик j-го критерия

Таблица 3.4.2 – Весовые коэффициенты оценки качества

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Фактор | Оценка | Код критерия | Оценка | Код | Наименование | Весовой коэффициент метрики |
| Надежность | 0,08 | 1.1 | 0,02 | Н01 | Средства восстановления при ошибках на входе | 0,02 |
| Н02 | Средства восстановления при сбоях оборудования | 0 |
| Н03 | Реализация управления средствами восстановления | 0 |
| 1.2 | 0,06 | Н04 | Функционирование в заданных режимах | 0,03 |
| Н05 | Обеспечение обработки заданного объема информации | 0,03 |
| Сопровожда-емость | 0,07 | 2.1 | 0,03 | С05 | Использование основных логических структур | 0,02 |
| С06 | Соблюдение принципа нисходящего программмирования | 0,01 |
| С07 | Комментарии обоснования декомпозиции программ при кодировании | 0 |
| 2.2 | 0,02 | С02 | Сложность архитектуры проекта | 0,01 |
| С10 | Простота кодирования | 0,01 |
| 2.3 | 0,01 | С04 | Принятая система идентификации | 0,01 |
| С08 | Комментарии логики | 0 |
| 2.4 | 0,01 | С09 | Оформление текста программ | 0,01 |
| Удобство применения | 0,32 | 3.1 | 0,09 | У01 | Освоение работы ПС | 0,07 |
| У02 | Документация для освоения | 0,01 |
| У03 | Полнота пользовательской документации | 0,01 |
| 3.2 | 0,1 | У04 | Точность пользовательской документации | 0,03 |
| У05 | Понятность пользовательской документации | 0,04 |

*Продолжение таблицы 3.4.2*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Фактор | Оценка | Код критерия | Оценка | Код | Наименование | Весовой коэффициент метрики |
| Удобство применения | 0,32 | 3.2 | 0,1 | У06 | Техническое использование пользовательской документации | 0,03 |
| У07 | Прослеживание вариантов пользовательской документации | 0 |
| 3.3 | 0,13 | У08 | Эксплуатация | 0,06 |
| У09 | Управление меню | 0,05 |
| У10 | Функция HELP | 0 |
| У11 | Управление данными | 0,02 |
| У12 | Рабочие процедуры | 0,01 |
| Эффектив-ность | 0,07 | 4.1 | 0,02 | Э01 | Уровень автоматизации | 0,02 |
| 4.2 | 0,04 | Э02 | Временная эффективность | 0,04 |
| 4.3 | 0,01 | Э03 | Ресурсоемкость | 0,01 |
| Универсаль-ность | 0,09 | 5.1 | 0,04 | Г01 | Широта охвата функций | 0,02 |
| Г02 | Простота архитектуры проекта | 0,01 |
| Г03 | Сложность архитектуры проекта | 0 |
| Г04 | Сложность структуры кода программ | 0 |
| Г05 | Применение стандартных потоков связи | 0,01 |
| Г06 | Применение стандартных интерфейсных программ | 0 |
| 5.2 | 0,01 | Г07 | Зависимость от используемого комплекса технических средств | 0 |
| Г08 | Зависимость от базового ПО | 0,01 |
| Г09 | Изоляция немобильности | 0 |
| 5.3 | 0,04 | Г10 | Простота кодирования | 0,01 |
| Г11 | Число комментариев | 0,01 |
| Г12 | Количество комментариев | 0,01 |

*Продолжение таблицы 3.4.2*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Фактор | Оценка | Код критерия | Оценка | Код | Наименование | Весовой коэффициент метрики |
| Универсаль-ность | 0,09 | 5.3 | 0,04 | Г13 | Использование описательных средств языка | 0,01 |
| Г14 | Независимость модулей | 0 |
| Корректность | 0,37 | 6.1 | 0,08 | К01 | Полнота документации разработки | 0,05 |
| К02 | Полнота программной документации | 0,03 |
| 6.2 | 0,23 | К03 | Непротиворечивость документации | 0,02 |
| К04 | Непротиворечивость программы | 0,01 |
| К05 | Единообразие интерфейсов между модулями и пользователем | 0,07 |
| К06 | Единообразие кодирования и определения переменных | 0,06 |
| К07 | Соответствие документации стандартам | 0,01 |
| К08 | Соответствие ПС стандартам программирования | 0,06 |
| 6.4 | 0 | К10 | Полнота тестирования проекта | 0 |
| 6.3 | 0,06 | К11 | Реализации всех решений | 0,03 |
| К12 | Отсутствие явных ошибок и достаточность реквизитов | 0,03 |

Таблица 3.4.3 – Оценочные элементы фактора «Надежность»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Н0101 | Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных | экспертный | 0,007 |
| Н0102 | Возможность обработки ошибочных ситуаций | экспертный | 0,01 |
| Н0103 | Полнота обработки ошибочных ситуаций | экспертный | 0,003 |
| Н0104 | Наличие тестов для проверки допустимых значений входных данных | экспертный | 0 |
| Н0105 | Наличие системы контроля полноты, входных данных | экспертный | 0 |
| Н0106 | Наличие средств контроля корректности входных данных | экспертный | 0 |

*Продолжение таблицы 3.4.3*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Н0107 | Наличие средств контроля непротиворечивости входных данных | экспертный | 0 |
| Н0201 | Наличие требований к программе по восстановлению процесса выполнения в случае сбоя операционной системы, процессора, внешних устройств | экспертный | 0 |
| Н0202 | Наличие требований к программе по восстановлению результатов при отказах процессора, ОС | экспертный | 0 |
| Н0203 | Наличие средств восстановления процесса в случае сбоев оборудования | экспертный | 0 |
| Н0204 | Наличие возможности разделения по времени выполнения отдельных функций программ | экспертный | 0 |
| Н0205 | Наличие возможности повторного старта с точки останова | экспертный | 0 |
| Н0108 | Наличие проверки параметров и адресов по диапазону их значений | экспертный | 0 |
| Н0109 | Наличие обработки граничных результатов | экспертный | 0 |
| Н0110 | Наличие обработки неопределенностей | экспертный | 0 |
| Н0301 | Наличие централизованного управления процессами, конкурирующими из-за ресурсов | экспертный | 0 |
| Н0302 | Наличие возможности автоматически обходить ошибочные ситуации в процессе вычисления | экспертный | 0 |
| Н0303 | Наличие средств, обеспечивающих завершение процесса решения в случае помех | экспертный | 0 |
| Н0304 | Наличие средств, обеспечивающих выполнение программы в сокращенном объеме в случае ошибок или помех | экспертный | 0 |
| Н0305 | Показатель устойчивости к искажающим воздействиям | расчетный | 0 |
| Н0401 | Вероятность безотказной работы | расчетный | 0,03 |
| Н0501 | Оценка по среднему времени восстановления | расчетный | 0,01 |
| Н0502 | Оценка по продолжительности преобразования входного набора данных в выходной | расчетный | 0,02 |

Таблица 3.4.4 – Оценочные элементы фактора «Сопровождаемость»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| С0803 | Наличие комментариев в точках входа и выхода программы | экспертный | 0 |
| С0302 | Оценка простоты программы по числу точек входа и выхода | расчетный | 0 |
| С1002 | Оценка простоты программы по числу переходов по условию | расчетный | 0 |
| С0303 | Осуществляется ли передача результатов работы модуля через вызывающий его модуль | экспертный | 0 |

*Продолжение таблицы 3.4.4*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| С0304 | Осуществляется ли контроль за правильностью данных, поступающих в вызывающий модуль от вызываемого | экспертный | 0 |
| С0604 | Оценка программы по числу циклов | экспертный | 0,002 |
| С0801 | Наличие комментариев ко всем машинно-зависимым частям программы | экспертный | 0 |
| С0802 | Наличие комментариев к машинно-зависимым операторам программы | экспертный | 0 |
| С0901 | Соответствие комментариев принятым соглашениям | экспертный | 0 |
| С1001 | Используется ли язык высокого уровня | экспертный | 0,01 |
| С0301 | Наличие проверки корректности передаваемых данных | экспертный | 0 |
| С0902 | Наличие комментариев-заголовков программы с указанием ее структурных и функциональных характеристик | экспертный | 0,003 |
| С0601 | Использование при построении программ метода структурного программирования | экспертный | 0,001 |
| С0602 | Соблюдение принципа разработки программы сверху вниз | экспертный | 0,006 |
| С0201 | Наличие ограничений на размеры модуля | экспертный | 0,01 |
| С0101 | Наличие модульной схемы программы | экспертный | 0 |
| С030 | Наличие требований к независимости модулей программы от типов и форматов выходных данных | экспертный | 0 |
| С0102 | Оценка программы по числу уникальных модулей | экспертный | 0 |
| С0903 | Оценка ясности и точности описания последовательности функционирования всех элементов программы | экспертный | 0,007 |
| С0603 | Оценка программы по числу циклов с одним входом и одним выходом | экспертный | 0,001 |

Таблица 3.4.5 – Оценочные элементы фактора «Удобство применения»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| У0101 | Возможность освоения программных средств по документации | экспертный | 0,0035 |
| У0102 | Возможность освоения ПС на контрольном примере при помощи ЭВМ | экспертный | 0 |
| У0103 | Возможность поэтапного освоения ПС | экспертный | 0,0035 |
| У0201 | Полнота и понятность документации для освоения | экспертный | 0,006 |
| У0202 | Точность документации для освоения | экспертный | 0,004 |

*Продолжение таблицы 3.4.5*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| У0203 | Техническое исполнение документации | экспертный | 0 |
| У0301 | Наличие краткой аннотации | экспертный | 0,002 |
| У0302 | Наличие описания решаемых задач | экспертный | 0,004 |
| У0303 | Наличие описания структуры функции ПС | экспертный | 0,001 |
| У0304 | Наличие описания основных функций ПС | экспертный | 0 |
| У0306 | Наличие описания частных функции | экспертный | 0 |
| У0307 | Наличие описания алгоритмов | экспертный | 0 |
| У0308 | Наличие описания межмодульных интерфейсов | экспертный | 0 |
| У0309 | Наличие описания пользовательских интерфейсов | экспертный | 0,001 |
| У0310 | Наличие описания входных и выходных данных | экспертный |  |
| У0311 | Наличие описания диагностических сообщений | экспертный | 0 |
| У0312 | Наличие описания основных характеристик ПС | экспертный | 0,001 |
| У0314 | Наличие описания программной среды функционирования ПС | экспертный | 0,001 |
| У0315 | Достаточность документации для ввода ПС в эксплуатацию | экспертный | 0 |
| У0316 | Наличие информации технологии перекоса для Мобильных программ | экспертный | 0 |
| У0401 | Соответствие оглавления содержанию документации | экспертный | 0 |
| У0402 | Оценка оформления документации | экспертный | 0 |
| У0403 | Грамматическая правильность изложения документации | экспертный | 0,007 |
| У0404 | Отсутствие противоречий | экспертный | 0,003 |
| У0405 | Отсутствие неправильных ссылок | экспертный | 0 |
| У0406 | Ясность формулировок и описаний | экспертный | 0,008 |
| У0407 | Отсутствие неоднозначных формулировок и описаний | экспертный | 0,002 |
| У0408 | Правильность использования терминов | экспертный | 0,005 |
| У0409 | Краткость, отсутствие лишней детализации | экспертный | 0 |
| У0410 | Единство формулировок | экспертный | 0 |
| У0411 | Единство обозначений | экспертный | 0 |
| У0412 | Отсутствие ненужных повторений | экспертный | 0 |

*Продолжение таблицы 3.4.5*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| У0413 | Наличие нужных объяснений | экспертный | 0,005 |
| У0501 | Оценка стиля изложения | экспертный | 0,02 |
| У0502 | Дидактическая разделенность | экспертный | 0,01 |
| У0503 | Формальная разделенность | экспертный | 0 |
| У0504 | Ясность логической структуры | экспертный | 0 |
| У0505 | Соблюдение стандартов и правил изложения в документации | экспертный | 0,01 |
| У0506 | Оценка по числу ссылок вперед в тексте документов | экспертный | 0 |
| У0601 | Наличие оглавления | экспертный | 0 |
| У0602 | Наличие предметного указателя | экспертный | 0 |
| У0603 | Наличие перекрестных ссылок | экспертный | 0 |
| У0604 | Наличие всех требуемых разделов | экспертный | 0 |
| У0605 | Соблюдение непрерывности нумерации страниц документов | экспертный | 0,01 |
| У0606 | Отсутствие незаконченных разделов абзацев, предложений | экспертный | 0,01 |
| У0607 | Наличие всех рисунков, чертежей, формул, таблиц | экспертный | 0,005 |
| У0608 | Наличие всех строк и примечаний | экспертный | 0,005 |
| У0609 | Логический порядок частей внутри главы | экспертный | 0 |
| У0701 | Наличие полного перечня документации | экспертный | 0 |
| У0801 | Уровень языка общения пользователя с программой | экспертный | 0,01 |
| У0802 | Легкость и быстрота загрузки и запуска программы | экспертный | 0,02 |
| У0803 | Легкость и быстрота завершения работы программы | экспертный | 0,02 |
| У0804 | Возможность распечатки содержимого программы | экспертный | 0 |
| У0805 | Возможность приостановки и повторного запуска работы без потерь информации | экспертный | 0 |
| У0901 | Соответствие меню требованиям пользователя | экспертный | 0,03 |
| У0902 | Возможность прямого перехода вверх и вниз по многоуровневому меню (пропуск уровней) | экспертный | 0,02 |
| У1001 | Возможность управления подробностью получаемых выходных данных | экспертный | 0 |
| У1002 | Достаточность полученной информации для продолжения работы | экспертный | 0 |

*Продолжение таблицы 3.4.5*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| У1101 | Обеспечение удобства ввода данных | экспертный | 0,01 |
| У1102 | Легкость восприятия | экспертный | 0,01 |
| У1201 | Обеспечение программой выполнения предусмотренных рабочих процедур | экспертный | 0,01 |
| У1202 | Достаточность информации, выдаваемой программой для составления дополнительных процедур | экспертный | 0 |

Таблица 3.4.6 – Оценочные элементы фактора «Эффективность»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Э0101 | Проблемно-ориентированные функции | экспертный или расчетный | 0,008 |
| Э0102 | Машинно-ориентированные функции | экспертный или расчетный | 0 |
| Э0103 | Функции ведения и управления | экспертный или расчетный | 0,002 |
| Э0104 | Функции ввода/вывода | экспертный или расчетный | 0,01 |
| Э0105 | Функции защиты и проверки данных | экспертный или расчетный | 0 |
| Э0106 | Функции защиты от несанкционированного доступа | экспертный или расчетный | 0 |
| Э0107 | Функции контроля доступа | экспертный или расчетный | 0 |
| Э0108 | Функции защиты от внесения изменений | экспертный или расчетный | 0 |
| Э0109 | Наличие соответствующих границ функциональных областей | экспертный или расчетный | 0 |
| Э0110 | Число знаков после запятой в результатах вычислений | экспертный или расчетный | 0 |
| Э0201 | Время выполнения программ | экспертный или расчетный | 0,01 |
| Э0202 | Время реакции и ответов | экспертный или расчетный | 0,01 |
| Э0203 | Время подготовки | экспертный или расчетный | 0,01 |
| Э0205 | Затраты времени на защиту данных | экспертный или расчетный | 0 |
| Э0206 | Время компиляции | экспертный или расчетный | 0,01 |
| Э0301 | Требуемый объем внутренней памяти | экспертный или расчетный | 0,001 |
| Э0302 | Требуемый объем внешней памяти | экспертный или расчетный | 0,001 |
| Э0303 | Требуемые периферийные устройства | экспертный или расчетный | 0,001 |

*Продолжение таблицы 3.4.6*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Э0304 | Требуемое базовое программное обеспечение | экспертный или расчетный | 0,007 |

Таблица 3.4.7 – Оценочные элементы фактора «Универсальность»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Г0101 | Оценка числа потенциальных пользователей | экспертный | 0,005 |
| Г0102 | Оценка числа функций ПС | экспертный | 0.003 |
| Г0103 | Насколько набор функций удовлетворяет требованиям пользователя | экспертный | 0,005 |
| Г0104 | Насколько возможности программ охватывают область решаемых пользователем задач | экспертный | 0,007 |
| Г0105 | Возможность настройки формата выходных данных для конкретных пользователей | экспертный | 0 |
| Г0201 | Наличие схемы иерархии модулей программы | экспертный | 0 |
| Г0202 | Оценка независимости модулей | экспертный | 0 |
| Г0203 | Оценка числа уникальных элементов/реквизитов | экспертный | 0 |
| Г0204 | Используется ли в текущем вызове модуля информация, полученная в предыдущем вызове | экспертный | 0 |
| Г0205 | Оценка организации точек входа и выхода модуля | экспертный | 0,01 |
| Г0206 | Наличие описания атрибутов модуля | экспертный | 0 |
| Г0301 | Оценка программ по числу переходов и точек ветвления | экспертный | 0 |
| Г0401 | Использование метода пошагового уточнения | экспертный | 0 |
| Г0402 | Наличие описания структуры программ | экспертный | 0 |
| Г0403 | Наличие описания связей между элементами структуры программы | экспертный | 0 |
| Г0404 | Наличие в программе повторного выполнения функций (подпрограмм) | экспертный | 0 |
| Г0501 | Использование стандартных протоколов связи | экспертный | 0,01 |
| Г0601 | Использование стандартных интерфейсных подпрограмм | экспертный | 0 |
| Г0701 | Оценка зависимости программ от емкости оперативной памяти ЭВМ | экспертный | 0 |
| Г0702 | Оценка зависимости временных характеристик программы от скорости вычислении ЭВМ | экспертный | 0 |
| Г0703 | Оценка зависимости функционирования программы от числа внешних запоминающих устройств и их общей емкости | экспертный | 0 |

*Продолжение таблицы 3.4.7*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Г0704 | Оценка зависимости функционирования программы от специальных устройств ввода-вывода | экспертный | 0 |
| Г0801 | Применение специальных языков программирования | экспертный | 0 |
| Г0802 | Оценка зависимости программы от программ операционной системы | экспертный | 0,007 |
| Г0803 | Зависимость от других программных средств | экспертный | 0,003 |
| Г0901 | Оценка локализации непереносимой части программы | экспертный | 0 |
| Г1001 | Оценка использования отрицательных или булевых выражений | экспертный | 0 |
| Г1002 | Оценка программы по использованию условных переходов | экспертный | 0 |
| Г1003 | Оценка программы по использованию безусловных переходов | экспертный | 0 |
| Г1004 | Оформление процедур входа и выхода из циклов | экспертный | 0,005 |
| Г1005 | Ограничения на модификацию переменной индексации в цикле | экспертный | 0 |
| Г1007 | Оценка программы по использованию локальных переменных | экспертный | 0,005 |
| Г1006 | Оценка модулей по направлению потока управления | экспертный | 0 |
| Г1101 | Оценка программы по числу комментариев | экспертный | 0,01 |
| Г1201 | Наличие заголовка в программе | экспертный | 0,01 |
| Г1202 | Комментарии к точкам ветвлений | экспертный | 0 |
| Г1203 | Комментарии к машинно-зависимым частям программы | экспертный | 0 |
| Г1204 | Комментарии к машинно-зависимым операторам программы | экспертный | 0 |
| Г1205 | Комментарии к операторам объявления переменных | экспертный | 0 |
| Г1206 | Оценка семантики операторов | экспертный | 0 |
| Г1207 | Наличие соглашений по форме представления комментариев | экспертный | 0 |
| Г1208 | Наличие общих комментариев к программам | экспертный | 0 |
| Г1301 | Использование языков высокого уровня | экспертный | 0,01 |
| Г1302 | Семантика имен используемых переменных | экспертный | 0 |
| Г1303 | Использование отступов, сдвигов и пропусков при формировании текста | экспертный | 0 |
| Г1304 | Размещение операторов по строкам | экспертный | 0 |

*Продолжение таблицы 3.4.7*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| Г1401 | Передача информации для управления по параметрам | экспертный | 0 |
| Г1402 | Параметрическая передача входных данных | экспертный |  |
| Г1403 | Наличие передачи результатов работы между модулями | экспертный | 0 |
| Г0404 | Наличие проверки правильности данных, получаемых модулями от вызываемого модуля | экспертный | 0 |
| Г1405 | Использование общих областей памяти | экспертный | 0 |

Таблица 3.4.8 – Оценочные элементы фактора «Универсальность»

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| К0101 | Наличие всех необходимых документов для понимания и использования ПС | экспертный | 0,02 |
| К0102 | Наличие описания и схемы иерархии модулей программы | экспертный | 0,01 |
| К0103 | Наличие описания основных функций | экспертный | 0 |
| К0104 | Наличие описания частных функций | экспертный | 0 |
| К0105 | Наличие описания данных | экспертный | 0 |
| К0106 | Наличие описания алгоритмов | экспертный | 0 |
| К0107 | Наличие описания интерфейсов между модулями | экспертный | 0 |
| К0108 | Наличие описания интерфейсов с пользователями | экспертный | 0,02 |
| К0109 | Наличие описания используемых числовых методов | экспертный | 0 |
| К0110 | Указаны ли все численные методы | экспертный | 0 |
| К0111 | Наличие описания всех параметров | экспертный | 0 |
| К0112 | Наличие описания методов настройки системы | экспертный | 0 |
| К0113 | Наличие описания всех диагностических сообщений | экспертный | 0 |
| К0114 | Наличие описания способов проверки работоспособности программы | экспертный | 0 |
| К0201 | Реализация всех исходных модулей | экспертный | 0 |
| К0202 | Реализация всех основных функций | экспертный | 0 |
| К0203 | Реализация всех частных функций | экспертный | 0 |
| К0204 | Реализация всех алгоритмов | экспертный | 0 |
| К0205 | Реализация всех взаимосвязей в системе | экспертный | 0,01 |

*Продолжение таблицы 3.4.8*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| К0206 | Реализация всех интерфейсов между модулями | экспертный | 0,01 |
| К0207 | Реализация возможности настройки системы | экспертный | 0 |
| К0208 | Реализация диагностики всех граничных и аварийных ситуаций | экспертный | 0 |
| К0209 | Наличие определения всех данных (переменные, индексы, массивы и проч.) | экспертный | 0 |
| К0210 | Наличие интерфейсов с пользователем | экспертный | 0,01 |
| К0301 | Отсутствие противоречий в описания частных функций | экспертный | 0 |
| К0302 | Отсутствие противоречий в описания основных функций в разных документах | экспертный | 0 |
| К0303 | Отсутствие противоречий в описания алгоритмов | экспертный | 0 |
| К0304 | Отсутствие противоречий в описания взаимосвязей в системе | экспертный | 0,01 |
| К0305 | Отсутствие противоречий в описания интерфейсов между модулями | экспертный | 0 |
| К0306 | Отсутствие противоречий в описания интерфейсов с пользователем | экспертный | 0,01 |
| К0307 | Отсутствие противоречий в описания настройки системы | экспертный | 0 |
| К0309 | Отсутствие противоречий в описания иерархической структуры сообщений | экспертный | 0 |
| К0310 | Отсутствие противоречий в описания диагностических сообщений | экспертный | 0 |
| К0311 | Отсутствие противоречий в описания данных | экспертный | 0 |
| К0401 | Отсутствие противоречий в выполнении основных функций | экспертный | 0 |
| К0402 | Отсутствие противоречий в выполнении частных функций | экспертный | 0 |
| К0403 | Отсутствие противоречий в выполнении алгоритмов | экспертный | 0 |
| К0404 | Правильность взаимосвязей | экспертный | 0,005 |
| К0405 | Правильность реализации интерфейса между модулями | экспертный | 0 |
| К0406 | Правильность реализации интерфейса с пользователем | экспертный | 0,005 |
| К0407 | Отсутствие противоречий в настройке системы | экспертный | 0 |
| К0408 | Отсутствие противоречий в диагностике системы | экспертный | 0 |
| К0409 | Отсутствие противоречий в общих переменных | экспертный | 0 |
| К0501 | Единообразие способов вызова модулей | экспертный | 0,02 |

*Продолжение таблицы 3.4.8*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| К0502 | Единообразие процедур возврата управления из модулей | экспертный | 0,02 |
| К0503 | Единообразие способов сохранения информации для возврата | экспертный | 0 |
| К0504 | Единообразие способов восстановления информации для возврата | экспертный | 0 |
| К0505 | Единообразие организации списков передаваемых параметров | экспертный | 0,03 |
| К0601 | Единообразие наименования каждой переменной и константы | экспертный | 0,02 |
| К0602 | Все ли одинаковые константы встречаются во всех программах под одинаковыми именами | экспертный | 0,01 |
| К0603 | Единообразие определения внешних данных во всех программах | экспертный | 0,02 |
| К0604 | Используются ли разные идентификаторы для разных переменных | экспертный | 0,01 |
| К0605 | Все ли общие переменные объявлены как общие переменные | экспертный | 0 |
| К0606 | Наличие определений одинаковых атрибутов | экспертный | 0 |
| К0701 | Комплектность документации в соответствии со стандартами | экспертный | 0,01 |
| К0702 | Правильное оформление частей документов | экспертный | 0 |
| К0703 | Правильное оформление титульных и заглавных листов документов | экспертный | 0 |
| К0704 | Наличие в документах всех разделов в соответствии со стандартами | экспертный | 0 |
| К0705 | Полнота содержания разделов в соответствии со стандартами | экспертный | 0 |
| К0706 | Деление документов на структурные элементы: разделы, подразделы, пункты, подпункты | экспертный | 0 |
| К0801 | Соответствие организации и вычислительного процесса эксплуатационном документации | экспертный | 0,01 |
| К0802 | Правильность заданий на выполнение программы, правильность написания управляющих и операторов (отсутствие ошибок) | экспертный | 0 |
| К0803 | Отсутствие ошибок в описании действий пользователя | экспертный |  |
| К0804 | Отсутствие ошибок в описании запуска | экспертный | 0,02 |
| К0805 | Отсутствие ошибок в описании генерации | экспертный | 0,02 |
| К0806 | Отсутствие ошибок в описании настройки | экспертный | 0,01 |
| К1001 | Наличие требований к тестированию программ | экспертный | 0 |

*Продолжение таблицы 3.4.8*

|  |  |  |  |
| --- | --- | --- | --- |
| Код элемента | Наименование | Метод оценки | Оценка |
| К1002 | Достаточность требований к тестированию программ | экспертный | 0 |
| К1003 | Отношение числа модулей, отработавших в процессе тестирования и отладки () к общему числу модулей ( ) | расчетный | 0 |
| К1004 | Отношение числа логических блоков, отработавших в процессе тестирования и отладки (), к общему числу логических блоков в программе () | расчетный | 0 |

Вычисление итоговой оценки R-метрики j-критерия по формуле:

, (1)

где Q – число оценочных элементов в R-й метрике.

Таблица 3.4.9 – Итоговая оценка R-метрики j-критерия

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код | Наименование | Весовой коэффициент метрики | Число оценочных элементов |  |
| Н01 | Средства восстановления при ошибках на входе | 0,02 | 10 | 0.0020 |
| Н02 | Средства восстановления при сбоях оборудования | 0 | 5 | 0.0000 |
| Н03 | Реализация управления средствами восстановления | 0 | 5 | 0.0000 |
| Н04 | Функционирование в заданных режимах | 0,03 | 1 | 0.0300 |
| Н05 | Обеспечение обработки заданного объема информации | 0,03 | 2 | 0.0150 |
| С05 | Использование основных логических структур | 0,02 | 1 | 0.0200 |
| С06 | Соблюдение принципа нисходящего программмирования | 0,01 | 4 | 0.0025 |
| С07 | Комментарии обоснования декомпозиции программ при кодировании | 0 | 1 | 0.0000 |
| С02 | Сложность архитектуры проекта | 0,01 | 1 | 0.0100 |
| С10 | Простота кодирования | 0,01 | 2 | 0.0050 |
| С04 | Принятая система идентификации | 0,01 | 1 | 0.0100 |
| С08 | Комментарии логики | 0 | 3 | 0.0000 |

*Продолжение таблицы 3.4.9*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код | Наименование | Весовой коэффициент метрики | Число оценочных элементов |  |
| С09 | Оформление текста программ | 0,01 | 3 | 0.0033 |
| У01 | Освоение работы ПС | 0,07 | 3 | 0.0233 |
| У02 | Документация для освоения | 0,01 | 3 | 0.0033 |
| У03 | Полнота пользовательской документации | 0,01 | 14 | 0.0007 |
| У04 | Точность пользовательской документации | 0,03 | 13 | 0.0023 |
| У05 | Понятность пользовательской документации | 0,04 | 6 | 0.0067 |
| У06 | Техническое использование пользовательской документации | 0,03 | 9 | 0.0033 |
| У07 | Прослеживание вариантов пользовательской документации | 0 | 1 | 0.0000 |
| У08 | Эксплуатация | 0,06 | 5 | 0.0120 |
| У09 | Управление меню | 0,05 | 2 | 0.0250 |
| У10 | Функция HELP | 0 | 2 | 0.0000 |
| У11 | Управление данными | 0,02 | 2 | 0.0100 |
| У12 | Рабочие процедуры | 0,01 | 2 | 0.0050 |
| Э01 | Уровень автоматизации | 0,02 | 10 | 0.0020 |
| Э02 | Временная эффективность | 0,04 | 5 | 0.0080 |
| Э03 | Ресурсоемкость | 0,01 | 4 | 0.0025 |
| Г01 | Широта охвата функций | 0,02 | 5 | 0.0040 |
| Г02 | Простота архитектуры проекта | 0,01 | 6 | 0.0017 |
| Г03 | Сложность архитектуры проекта | 0 | 1 | 0.0000 |
| Г04 | Сложность структуры кода программ | 0 | 4 | 0.0000 |
| Г05 | Применение стандартных потоков связи | 0,01 | 1 | 0.0100 |
| Г06 | Применение стандартных интерфейсных программ | 0 | 1 | 0.0000 |
| Г07 | Зависимость от используемого комплекса технических средств | 0 | 4 | 0.0000 |
| Г08 | Зависимость от базового ПО | 0,01 | 3 | 0.0033 |
| Г09 | Изоляция немобильности | 0 | 1 | 0.0000 |

*Продолжение таблицы 3.4.9*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Код | Наименование | Весовой коэффициент метрики | Число оценочных элементов |  |
| Г10 | Простота кодирования | 0,01 | 7 | 0.0014 |
| Г11 | Число комментариев | 0,01 | 1 | 0.0100 |
| Г12 | Количество комментариев | 0,01 | 8 | 0.0013 |
| Г13 | Использование описательных средств языка | 0,01 | 4 | 0.0025 |
| Г14 | Независимость модулей | 0 | 5 | 0.0000 |
| К01 | Полнота документации разработки | 0,05 | 14 | 0.0036 |
| К02 | Полнота программной документации | 0,03 | 10 | 0.0030 |
| К03 | Непротиворечивость документации | 0,02 | 10 | 0.0020 |
| К04 | Непротиворечивость программы | 0,01 | 9 | 0.0011 |
| К05 | Единообразие интерфейсов между модулями и пользователем | 0,07 | 5 | 0.0140 |
| К06 | Единообразие кодирования и определения переменных | 0,06 | 6 | 0.0100 |
| К07 | Соответствие документации стандартам | 0,01 | 6 | 0.0017 |
| К08 | Соответствие ПС стандартам программирования | 0,06 | 6 | 0.0100 |
| К10 | Полнота тестирования проекта | 0 | 4 | 0.0000 |
| К11 | Реализации всех решений | 0,03 | 1 | 0.0300 |
| К12 | Отсутствие явных ошибок и достаточность реквизитов | 0,03 | 1 | 0.0300 |

Вычисление абсолютных показателей критериев i-фактора качества по формуле:

, (2)

где n – число метрик, относящихся к j-му критерию.

Фактор качества вычисляется по формуле:

, (3)

где .

Таблица 3.4.10 – Абсолютные показатели критериев i-фактора качества и фактор качества

|  |  |  |  |
| --- | --- | --- | --- |
| Фактор | Критерий |  |  |
| Надежность | 1.1 | 0,00004 | 0,00004 |
| 1.2 | 0,00135 | 0,00135 |
| Сопровождаемость | 2.1 | 0,000425 | 0,000425 |
| 2.2 | 0,00015 | 0,00015 |
| 2.3 | 0,0001 | 0,0001 |
| 2.4 | 0,000033 | 0,000033 |
| Удобство применения | 3.1 | 0,001671 | 0,001671 |
| 3.2 | 0,000436 | 0,000436 |
| 3.3 | 0,00222 | 0,00222 |
| Эффективность | 4.1 | 0,00004 | 0,00004 |
| 4.2 | 0,00032 | 0,00032 |
| 4.3 | 0,000025 | 0,000025 |
| Универсальность | 5.1 | 0,000197 | 0,000197 |
| 5.2 | 0,000033 | 0,000033 |
| 5.3 | 0,000152 | 0,000152 |
| Корректность | 6.1 | 0,00027 | 0,00027 |
| 6.2 | 0,002248 | 0,002248 |
| 6.3 | 0,0018 | 0,0018 |
| 6.4 | 0 | 0 |

По итогам анализа качества разработанной системы «АТО», согласно ГОСТ 28195-89, следует сделать вывод о том, что, данная система соответствует минимальному количеству метрик факторов качества программного средства, а значит, является низким по качеству программным продуктом для класса программ своего уровня.

3.5 Выводы по разделу

В этом разделе курсовой работы реализовывалась разработанная система на программном уровне. Для наглядного представления этого процесса были построены следующие диаграммы:

* диаграмма компонентов, которая описывает организацию и подключение физических компонентов в разработанной системе;
* диаграмма классов, которая дает обзор разработанной системы;
* граф диалога, который выявляет и устанавливает возможные тупиковые ситуации, выбирает рациональный путь перехода из текущего состояния системы в требуемое.

Кроме этого в этом разделе осуществлялось тестирование двумя методами: структурное и функциональное тестирование. Эти методы тестирования делились на подпункты и тестировались по отдельности.

Структурное тестирование:

* тестирование базового пути;
* тестирование условий;
* тестирование циклов.

Функциональное тестирование:

* разбиение на классы эквивалентности;
* анализ граничных значений;
* анализ причинно-следственных связей.

Кроме всего прочего рассчитывалась оценка качества программах средств согласно с ГОСТ 28195-89.

ЗАКЛЮЧЕНИЕ

В данной курсовой работе выполнен: анализ требований, проектирование и реализация программных средств, предоставляющих пользователю возможность реализовать все функции разрабатываемой системы. В результате выполнения работы была разработана система, автоматизирующее процесс продаж билетов клиентам. Приложение предоставляет пользователю следующие основные функции:

* авторизация пользователя;
* регистрация пользователя;
* поиск билетов для бронирования;
* бронирование билета;
* редактирование БД продаж авиабилетов
* личный кабинет пользователя, содержащий статистику по совершенным покупкам.

В ходе тестирования разработанной информационной системы было продемонстрировано, что система выполняет все поставленные задачи в соответствии со сформированными требованиями в разделе «Анализ технического задания», демонстрируя корректную работу.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. CoderLessons.com Функциональное тестирование [Электронный ресурс], URL: https://coderlessons.com/tutorials/kachestvo-programmnogo-obespecheniia/ruchnoe-testirovanie/funktsionalnoe-testirovanie-2 (дата обращения 15.05.2021)
2. Flexberry. Диаграмма последовательности (Sequence diagram) [Электронный ресурс], URL: https://flexberry.github.io/ru/fd\_sequence-diagram.html (дата обращения 09.05.2021)
3. GraphQL. Introduction to GraphQL [Электронный ресурс], URL: https://graphql.org/learn/ (дата обращения 03.05.2021)
4. Habr. UML для самых маленьких: диаграмма классов [Электронный ресурс], URL: https://habr.com/ru/post/511798/ (дата обращения 13.05.2021)
5. Habr. Клиент-серверная архитектура в картинках [Электронный ресурс], URL: https://habr.com/ru/post/495698/ (дата обращения 01.05.2021)
6. Interface Ltd. СРЕДСТВА СТРУКТУРНОГО ПРОЕКТИРОВАНИЯ [Электронный ресурс], URL: http://www.interface.ru/fset.asp?Url=/case/defs72.htm (дата обращения 10.05.2021)
7. PostgreSQL. Documentation [Электронный ресурс], URL: https://www.postgresql.org/docs/13/index.html (дата обращения 04.05.2021)
8. Redis. Memory Optimization [Электронный ресурс], URL: https://redis.io/topics/memory-optimization (дата обращения 02.05.2021)
9. Trinion. IDEF0. Знакомство с нотацией и пример использования. [Электронный ресурс], URL: https://trinion.org/articles/idef0-znakomstvo-s-notaciey-i-primer-ispolzovaniya/ (дата обращения 05.05.2021)
10. Алекс Дэвис Асинхронное программирование в C# 5.0. / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2013. – 120 с
11. Блог программиста. Блок-схемы алгоритмов. ГОСТ. Примеры [Электронный ресурс], URL: https://pro-prof.com/archives/1462 (дата обращения 11.05.2021)
12. Блог программиста. Основы UML — диаграммы использования (use-case) [Электронный ресурс], URL: https://pro-prof.com/archives/2594 (дата обращения 07.05.2021)
13. Диаграммы компонентов – Учебная и научная деятельность Анисимова [Электронный ресурс], URL: https://sites.google.com/site/anisimovkhv/learning/pris/lecture/tema15/tema15\_2 (дата обращения 12.05.2021)
14. Мюллер, Джон Пол, Семnф, Билл, Сфер, Чак. С# для чайников., Пер. с англ. – СПб.: ООО «Диалектика», 2019. – 608 с.
15. НОУ ИНТУИТ. Диаграмма активностей: крупным планом [Электронный ресурс], URL: https://intuit.ru/studies/courses/1007/229/lecture/5958 (дата обращения 08.05.2021)
16. Студопедия. Диаграммы переходов состояний [Электронный ресурс], URL: https://studepedia.org/index.php?vol=2&post=20754 (дата обращения 06.05.2021)
17. Студопедия. Структурное тестирование — Студопедия классов [Электронный ресурс], URL: https://studopedia.ru/3\_80803\_strukturnoe-testirovanie.html (дата обращения 14.05.2021)
18. Тюкачев Н. А., Хлебостроев В. Г. C#. Алгоритмы и структуры данных: Учебное пособие. — 3-е изд., стер. — СПб.: Издательство «Лань», 2018. — 232 с.
19. Фримен, Адам. ASP.NET Core MVC 2 с примерами на С# для профессионалов. 7-е изд., Пер. с англ. - СПб.: ООО «Диалектика», 2019. – 1008 с.
20. Электронный фонд правовых и нормативно-технических документов. ГОСТ 28195-89 Оценка качества программных средств. Общие положения - docs.cntd.ru [Электронный ресурс], URL: https://docs.cntd.ru/document/1200009135 (дата обращения 16.05.2021)

ПРИЛОЖЕНИЕ А – КОД ПРОГРАММЫ

Файл Startup.cs

**using** Microsoft**.**AspNetCore**.**Builder**;**

**using** Microsoft**.**AspNetCore**.**Hosting**;**

**using** Microsoft**.**AspNetCore**.**Http**;**

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** Microsoft**.**Extensions**.**Hosting**;**

**using** server**.**DB**;**

**using** Microsoft**.**EntityFrameworkCore**;**

**using** HotChocolate**;**

**using** Microsoft**.**Extensions**.**Configuration**;**

**using** Microsoft**.**AspNetCore**.**Authentication**.**JwtBearer**;**

**using** Microsoft**.**IdentityModel**.**Tokens**;**

**using** Microsoft**.**AspNetCore**.**Identity**;**

**using** server**.**Models**;**

**using** server**.**Schema**;**

**using** System**.**Text**;**

**using** server**.**Services**;**

**using** server**.**Middlewares**;**

**using** NetCasbin**;**

**using** NetCasbin**.**Persist**;**

**using** Casbin**.**Adapter**.**EFCore**;**

**using** server**.**Logger**;**

**namespace** server

**{**

**public** class Startup

**{**

**public** Startup**(**IConfiguration configuration**)**

**{**

Configuration **=** configuration**;**

**}**

**public** IConfiguration Configuration **{** **get;** **}**

// This method gets called by the runtime. Use this method to add services to the container.

**public** void ConfigureServices**(**IServiceCollection services**)**

**{**

//db connection

services**.**AddPooledDbContextFactory**<**ApplicationDbContext**>(**options **=>**

**{**

options**.**UseNpgsql**(**Configuration**.**GetConnectionString**(**"Default"**));**

**});**

//db connection for identity service

services**.**AddDbContext**<**ApplicationDbContext**>(**options **=>**

**{**

options**.**UseNpgsql**(**Configuration**.**GetConnectionString**(**"Default"**));**

**},**

contextLifetime**:** ServiceLifetime**.**Transient**,**

optionsLifetime**:** ServiceLifetime**.**Singleton

**);**

//identity service

var builder **=** services**.**AddIdentityCore**<**User**>(**opt **=>**

**{**

opt**.**Password**.**RequireNonAlphanumeric **=** **false;**

opt**.**Password**.**RequireUppercase **=** **false;**

**});**

builder **=** **new** IdentityBuilder**(**builder**.**UserType**,** **typeof(**IdentityRole**),** builder**.**Services**);**

builder**.**AddEntityFrameworkStores**<**ApplicationDbContext**>()**

**.**AddDefaultTokenProviders**()**

**.**AddRoleValidator**<**RoleValidator**<**IdentityRole**>>()**

**.**AddRoleManager**<**RoleManager**<**IdentityRole**>>()**

**.**AddSignInManager**<**SignInManager**<**User**>>();**

//auth

services**.**Configure**<**TokenSettings**>(**Configuration**.**GetSection**(**"TokenSettings"**));**

services**.**AddAuthentication**(**JwtBearerDefaults**.**AuthenticationScheme**)**

**.**AddJwtBearer**(**options **=>**

**{**

options**.**RequireHttpsMetadata **=** **false;**

options**.**TokenValidationParameters **=** **new** TokenValidationParameters**()**

**{**

ValidIssuer **=** Configuration**.**GetSection**(**"TokenSettings"**).**GetValue**<**string**>(**"Issuer"**),**

ValidateIssuer **=** **true,**

ValidAudience **=** Configuration**.**GetSection**(**"TokenSettings"**).**GetValue**<**string**>(**"Audience"**),**

ValidateAudience **=** **true,**

IssuerSigningKey **=** **new** SymmetricSecurityKey**(**Encoding**.**UTF8**.**GetBytes**(**Configuration**.**GetSection**(**"TokenSettings"**).**GetValue**<**string**>(**"Key"**))),**

ValidateIssuerSigningKey **=** **true**

**};**

**});**

services**.**AddScoped**<**ITokenService**,** TokenService**>();**

services**.**AddScoped**<**IIdentityService**,** IdentityService**>();**

services**.**AddScoped**<**TokenMiddleware**>();**

services**.**AddStackExchangeRedisCache**(**action **=>**

**{**

action**.**Configuration **=** Configuration**.**GetConnectionString**(**"Redis"**);**

**});**

//services.AddAuthorization();

//graphql

services**.**AddGraphQLServer**()**

**.**AddQueryType**<**Query**>()**

**.**AddMutationType**<**Mutation**>()**

**.**AddFiltering**()**

**.**AddSorting**()**

**.**AddProjections**()**

**.**AddDiagnosticEventListener**(**sp **=>**

**new** MiniProfilerQueryLogger**());**

services**.**AddMiniProfiler**(**options **=>**

**{** options**.**RouteBasePath **=** "/profiler"**;** **}).**AddEntityFramework**();**

//services.AddDbContext<ApplicationDbContext>(opt => opt.UseNpgsql(Configuration.GetConnectionString("Default")));

services**.**AddScoped**<**IAdapter**>(**x **=>** **new** EFCoreAdapter**<**int**>(**x**.**GetRequiredService**<**ApplicationDbContext**>()));**

//services.AddScoped(x => new Enforcer("./Casbin/auth\_model.conf", x.GetRequiredService<IAdapter>()));

**}**

**public** void Configure**(**IApplicationBuilder app**,** IWebHostEnvironment env**)**

**{**

**if** **(**env**.**IsDevelopment**())**

app**.**UseMiniProfiler**();**

app**.**UseRouting**();**

app**.**UseAuthentication**();**

//app.UseAuthorization();

app**.**UseMiddleware**<**TokenMiddleware**>();**

app**.**UseEndpoints**(**endpoints **=>**

**{**

endpoints**.**MapGraphQL**(**"/"**);**

**});**

**}**

**}**

**}**

Файл appsettings.Development.json

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"ConnectionStrings": {

"Default": "Host=localhost;Username=postgres;Password=010680;Database=ato",

"Redis": "localhost"

},

"TokenSettings": {

"Issuer": "localhost:5001",

"Audience": "API",

"Key": "MySecuredKey12345678910MySecuredKey12345678910"

}

}

Файл IdentityService.cs

**using** server**.**Models**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**using** Microsoft**.**AspNetCore**.**Identity**;**

**using** Microsoft**.**AspNetCore**.**Http**;**

**namespace** server**.**Services

**{**

**public** class IdentityService **:** IIdentityService

**{**

**private** **readonly** UserManager**<**User**>** \_userManager**;**

**private** **readonly** SignInManager**<**User**>** \_signInManager**;**

**private** **readonly** ITokenService \_tokenService**;**

**private** **readonly** IHttpContextAccessor \_httpContext**;**

**public** IdentityService**(**UserManager**<**User**>** userManager**,** SignInManager**<**User**>** signInManager**,** ITokenService tokenService**,** IHttpContextAccessor httpContext**)**

**{**

\_userManager **=** userManager**;**

\_signInManager **=** signInManager**;**

\_tokenService **=** tokenService**;**

\_httpContext **=** httpContext**;**

**}**

**public** **async** Task**<**IdentityResult**>** ChangePasswordAsync**(**User user**,** string currentPassword**,** string newPassword**)**

**{**

//revoke all tokens

**return** **await** \_userManager**.**ChangePasswordAsync**(**user**,** currentPassword**,** newPassword**);**

**}**

**public** **async** Task**<**User**>** GetCurrentUser**()**

**{**

**return** **await** \_userManager**.**FindByIdAsync**(**GetCurrentUserId**());** //\_tokenService.CurrentToken()?.User ?? throw new NullReferenceException("Please SignIn");

**}**

**public** string GetCurrentUserId**()**

**{**

**return** \_httpContext**.**HttpContext**?.**User**.**Claims**.**First**().**Value **??** string**.**Empty**;**

**}**

**public** **async** Task**<**Token**>** SignInAsync**(**string email**,** string password**)**

**{**

var user **=** **await** \_userManager**.**FindByEmailAsync**(**email**);**

var signInRes **=** **await** \_signInManager**.**CheckPasswordSignInAsync**(**user**,** password**,** **false);**

**if** **(**signInRes**.**Succeeded**)**

**{**

**return** **await** \_tokenService**.**CreateAsync**(**user**);**

**}**

**else**

**throw** **new** Exception**(**"Wrong username or password"**);**

**}**

**public** **async** Task SignOutAsync**()**

**{**

**await** \_tokenService**.**RevokeAsyncByAccess**(**\_tokenService**.**CurrentAccessToken**());**

**}**

**public** **async** Task**<**IdentityResult**>** SignUpAsync**(**string email**,** string password**)**

**{**

var user **=** **new** User**()** **{** Email **=** email**,** UserName **=** email **};**

var result **=** **await** \_userManager**.**CreateAsync**(**user**,** password**);**

**return** result**;**

**}**

**}**

**}**

Файл IIdentityService.cs

**using** Microsoft**.**AspNetCore**.**Identity**;**

**using** server**.**Models**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Services

**{**

**public** **interface** IIdentityService

**{**

Task**<**IdentityResult**>** SignUpAsync**(**string email**,** string password**);**

Task**<**Token**>** SignInAsync**(**string email**,** string password**);**

Task SignOutAsync**();**

string GetCurrentUserId**();**

Task**<**User**>** GetCurrentUser**();**

Task**<**IdentityResult**>** ChangePasswordAsync**(**User user**,** string currentPassword**,** string newPassword**);**

**}**

**}**

Файл ITokenService.cs

**using** System**;**

**using** System**.**Threading**.**Tasks**;**

**using** server**.**Models**;**

**namespace** server**.**Services

**{**

**public** **interface** ITokenService

**{**

Task**<**Token**>** UpdateAsync**(**string refreshToken**);**

Task RevokeAsyncByRefresh**(**string refreshToken**);**

Task RevokeAsyncByAccess**(**string accessToken**);**

Task**<**Token**>** CreateAsync**(**User user**);**

Task**<**bool**>** IsActive**(**string accessToken**);**

Token**?** CurrentToken**();**

string CurrentAccessToken**();**

**}**

**}**

Файл TokenService.cs

**using** Microsoft**.**IdentityModel**.**JsonWebTokens**;**

**using** server**.**Models**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**using** server**.**DB**;**

**using** System**.**Security**.**Claims**;**

**using** System**.**IdentityModel**.**Tokens**.**Jwt**;**

**using** Microsoft**.**Extensions**.**Options**;**

**using** Microsoft**.**IdentityModel**.**Tokens**;**

**using** System**.**Text**;**

**using** System**.**Security**.**Cryptography**;**

**using** Microsoft**.**Extensions**.**Caching**.**Distributed**;**

**using** Microsoft**.**AspNetCore**.**Http**;**

**using** Microsoft**.**Extensions**.**Primitives**;**

**namespace** server**.**Services

**{**

**public** class TokenService **:** ITokenService

**{**

**private** **readonly** ApplicationDbContext \_applicationDbContext**;**

**private** **readonly** IDistributedCache \_cache**;**

**private** **readonly** IOptions**<**TokenSettings**>** \_tokenSettings**;**

**private** **readonly** SymmetricSecurityKey \_symmetricSecurityKey**;**

**private** **readonly** IHttpContextAccessor \_httpContext**;**

**public** TokenService**(**ApplicationDbContext applicationDbContext**,** IOptions**<**TokenSettings**>** tokenSettings**,** IDistributedCache cache**,** IHttpContextAccessor httpContext**)**

**{**

\_applicationDbContext **=** applicationDbContext**;**

\_tokenSettings **=** tokenSettings**;**

\_symmetricSecurityKey **=** **new** SymmetricSecurityKey**(**Encoding**.**UTF8**.**GetBytes**(**\_tokenSettings**.**Value**.**Key**));**

\_cache **=** cache**;**

\_httpContext **=** httpContext**;**

**}**

**public** **async** Task**<**Token**>** CreateAsync**(**User user**)**

**{**

//access token generation

var identity **=** **new** ClaimsIdentity**(new[]** **{**

**new** Claim**(**ClaimTypes**.**NameIdentifier**,** user**.**Id**),**

**});**

var credentials **=** **new** SigningCredentials**(**\_symmetricSecurityKey**,** SecurityAlgorithms**.**HmacSha256**);**

var jwtToken **=** **new** JwtSecurityToken**(**

issuer**:** \_tokenSettings**.**Value**.**Issuer**,**

audience**:** \_tokenSettings**.**Value**.**Audience**,**

expires**:** DateTime**.**Now**.**AddMinutes**(**20**),**

signingCredentials**:** credentials**,**

claims**:** identity**.**Claims

**);**

string accessToken **=** **new** JwtSecurityTokenHandler**().**WriteToken**(**jwtToken**);**

//refresh token generation

var randomNumber **=** **new** byte**[**32**];**

**using** var rng **=** RandomNumberGenerator**.**Create**();**

rng**.**GetBytes**(**randomNumber**);**

var refreshToken **=** Convert**.**ToBase64String**(**randomNumber**);**

//Создание запси в бд

var token **=** **new** Token**()** **{** AccessToken **=** accessToken**,** RefreshToken **=** refreshToken**,** User\_id **=** user**.**Id **};**

//Сохраняем

**await** \_applicationDbContext**.**Tokens**.**AddAsync**(**token**);**

**await** \_applicationDbContext**.**SaveChangesAsync**();**

**return** token**;**

**}**

**public** **async** Task RevokeAsyncByRefresh**(**string refreshToken**)**

**{**

var token **=** \_applicationDbContext**.**Tokens**.**SingleOrDefault**(**x **=>** x**.**RefreshToken **==** refreshToken**);**

**if** **(**token **is** not **null** **&&** **!**token**.**Revoked**)**

**{**

token**.**Revoked **=** **true;**

**await** \_applicationDbContext**.**SaveChangesAsync**();**

//add access token to reddis blacklist

**await** DiactivateAsync**(**token**.**AccessToken**);**

**}**

**else**

**throw** **new** NullReferenceException**(**"Not valid token"**);**

**}**

**public** **async** Task RevokeAsyncByAccess**(**string accessToken**)**

**{**

var token **=** \_applicationDbContext**.**Tokens**.**SingleOrDefault**(**x **=>** x**.**AccessToken **==** accessToken**);**

**if** **(**token **is** not **null** **&&** **!**token**.**Revoked**)**

**{**

token**.**Revoked **=** **true;**

**await** \_applicationDbContext**.**SaveChangesAsync**();**

//add access token to reddis blacklist

**await** DiactivateAsync**(**token**.**AccessToken**);**

**}**

**else**

**throw** **new** NullReferenceException**(**"Not valid token"**);**

**}**

**public** **async** Task**<**Token**>** UpdateAsync**(**string refreshToken**)**

**{**

var token **=** \_applicationDbContext**.**Tokens**.**SingleOrDefault**(**x **=>** x**.**RefreshToken **==** refreshToken**);**

**if** **(**token **is** not **null** **&&** **!**token**.**Revoked**)**

**{**

token**.**Revoked **=** **true;**

**await** DiactivateAsync**(**token**.**AccessToken**);**

**return** **await** CreateAsync**(**token**.**User**);**

**}**

**else**

**throw** **new** NullReferenceException**(**"Not valid token"**);**

**}**

**public** **async** Task**<**bool**>** IsActive**(**string accessToken**)**

**{**

**return** **(await** \_cache**.**GetStringAsync**(**$"tokens:{accessToken}:deactivated"**))** **is** **null;**

**}**

**private** **async** Task DiactivateAsync**(**string accessToken**)**

**{**

**await** \_cache**.**SetStringAsync**(**$"tokens:{accessToken}:deactivated"**,** " "**,** **new** DistributedCacheEntryOptions**(){**

AbsoluteExpirationRelativeToNow **=** TimeSpan**.**FromMinutes**(**20**)**

**});**

**}**

**public** Token**?** CurrentToken**()**//todo: broken

**{**

var curr **=** CurrentAccessToken**();**

**return** \_applicationDbContext**.**Tokens**.**SingleOrDefault**(**x **=>** x**.**AccessToken**==**curr**);**

**}**

**public** string CurrentAccessToken**()**

**{**

var authorizationHeader **=** \_httpContext**.**HttpContext**?.**Request**?.**Headers**?[**"authorization"**];**

**return** authorizationHeader **is** **null** **||** authorizationHeader**.**Value **==** StringValues**.**Empty

**?** string**.**Empty

**:** authorizationHeader**.**Value**.**Single**().**Split**(**" "**).**Last**();**

**}**

**}**

**}**

Файл TokenSettings.cs

**namespace** server**.**Models

**{**

**public** class TokenSettings

**{**

**public** string Issuer **{** **get;** **set;** **}** **=** **default!;**

**public** string Audience **{** **get;** **set;** **}** **=** **default!;**

**public** string Key **{** **get;** **set;** **}** **=** **default!;**

**}**

**}**

Файл Air.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Air

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**10**)]**

**public** string BortNumber **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**10**)]**

**public** string Model **{** **get;** **set;** **}** **=** **default!;**

**[**Required**]**

**public** DateTime Date\_create **{** **get;** **set;** **}** **=** DateTime**.**Today**;**

**[**Required**,** StringLength**(**10**)]**

**public** int LifeTime **{** **get;** **set;** **}**

**[**Required**]**

**public** int Seats **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**5**)]**

**public** bool IsActive **{** **get;** **set;** **}** **=** **false;**

**[**Required**]**

**public** int SotrudnikId **{** **get;** **set;** **}**

**public** Sotrudnik Sotrudnik **{** **get;** **set;** **}** **=** **default!;**

**public** ICollection**<**Flight**>** Flights **{** **get;** **set;** **}** **=** **new** List**<**Flight**>();**

**}**

**}**

Файл Card.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Card

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**30**)]**

**public** string Name **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**5**)]**

**public** decimal Percent **{** **get;** **set;** **}**

**public** ICollection**<**Order**>** Orders **{** **get;** **set;** **}** **=** **new** List**<**Order**>();**

**}**

**}**

Файл Client.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Client

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**30**)]**

**public** string LastName **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Name **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string SurName **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**4**)]**

**public** int PassportSeia **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**6**)]**

**public** int PassportNumber **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**30**)]**

**public** string Addres **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Phone **{** **get;** **set;** **}** **=** **default!;**

**public** ICollection**<**Order**>** Orders **{** **get;** **set;** **}** **=** **new** List**<**Order**>();**

**}**

**}**

Файл Flight.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Flight

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**]**

**public** DateTime DateStart **{** **get;** **set;** **}** **=** DateTime**.**Today**;**

**[**Required**]**

**public** DateTime TimeStart **{** **get;** **set;** **}** **=** DateTime**.**Today**;**

**[**Required**]**

**public** bool IsActive **{** **get;** **set;** **}** **=** **false;**

**[**Required**]**

**public** int RouteId **{** **get;** **set;** **}**

**public** Route Route **{** **get;** **set;** **}** **=** **default!;**

**[**Required**]**

**public** int AirId **{** **get;** **set;** **}**

**public** Air Air **{** **get;** **set;** **}** **=** **default!;**

**public** ICollection**<**Order**>** Orders **{** **get;** **set;** **}** **=** **new** List**<**Order**>();**

**}**

**}**

Файл Order.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Order

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**30**)]**

**public** string Mesto **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**6**)]**

**public** int Counts **{** **get;** **set;** **}**

**[**Required**]**

**public** int FlightId **{** **get;** **set;** **}**

**public** Flight Flight **{** **get;** **set;** **}** **=** **default!;**

**[**Required**]**

**public** int ClientId **{** **get;** **set;** **}**

**public** Client Client **{** **get;** **set;** **}** **=** **default!;**

**[**Required**]**

**public** int CardId **{** **get;** **set;** **}**

**public** Card Card **{** **get;** **set;** **}** **=** **default!;**

**}**

**}**

Файл Route.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Route

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**30**)]**

**public** string Start **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Target **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**10**)]**

**public** decimal Price **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**5**)]**

**public** int Time **{** **get;** **set;** **}**

**public** ICollection**<**Flight**>** Flights **{** **get;** **set;** **}** **=** **new** List**<**Flight**>();**

**}**

**}**

Файл Sotrudnik.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Models

**{**

**public** class Sotrudnik

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**,** StringLength**(**30**)]**

**public** string Last\_name **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Name **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Sur\_name **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Addres **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**30**)]**

**public** string Phone **{** **get;** **set;** **}** **=** **default!;**

**[**Required**,** StringLength**(**3**)]**

**public** int Stage **{** **get;** **set;** **}**

**public** ICollection**<**Air**>** Airs **{** **get;** **set;** **}** **=** **new** List**<**Air**>();**

**}**

**}**

Файл Token.cs

**using** Microsoft**.**AspNetCore**.**Identity**;**

**using** System**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**namespace** server**.**Models

**{**

**public** class Token

**{**

**public** int Id **{** **get;** **set;** **}**

**[**Required**]**

**public** string RefreshToken **{** **get;** **set;** **}** **=** **default!;**

**[**Required**]**

**public** string AccessToken **{** **get;** **set;** **}** **=** **default!;**

**[**Required**]**

**public** bool Revoked **{** **get;** **set;** **}** **=** **false;**

**[**Required**]**

**public** DateTime CreatedAt **{** **get;** **set;** **}** **=** DateTime**.**Now**;**

**[**Required**]**

**public** DateTime UpdatedAt **{** **get;** **set;** **}** **=** DateTime**.**Now**;**

**[**Required**]**

**public** string User\_id **{** **get;** **set;** **}** **=** **default!;**

**public** User User **{** **get;** **set;** **}** **=** **default!;**

**}**

**}**

Файл User.cs

**using** HotChocolate**;**

**using** Microsoft**.**AspNetCore**.**Identity**;**

**using** System**.**ComponentModel**.**DataAnnotations**;**

**namespace** server**.**Models

**{**

**public** class User **:** IdentityUser

**{**

**public** string**?** Name **{** **get;** **set;** **}**

//[Required, StringLength(200)]

**public** string**?** Surname **{** **get;** **set;** **}**

//[Required, StringLength(200)]

**public** string**?** Patronymic **{** **get;** **set;** **}**

**[**GraphQLIgnore**]**

**public** **override** string PasswordHash **{** **get** **=>** **base.**PasswordHash**;** **set** **=>** **base.**PasswordHash **=** **value;** **}**

**}**

**}**

Файл LoginPayload.cs

**using** System**.**Collections**.**Generic**;**

**using** server**.**Models**;**

**namespace** server**.**Schema**.**Types

**{**

**public** class LoginPayload **:** Payload

**{**

**public** Token**?** Token **{** **get;** **set;** **}**

**public** LoginPayload**(**IReadOnlyList**<**Error**>** errors**,**

Token**?** token **=** **null,**

bool success **=** **true)** **:** **base(**errors**,** success**)**

**{**

Token **=** token**;**

**}**

**public** LoginPayload**(**Token**?** token **=** **null,**

bool success **=** **true)** **:** **base(**success**)**

**{**

Token **=** token**;**

**}**

**}**

**}**

Файл Payload.cs

**using** System**.**Collections**.**Generic**;**

**namespace** server**.**Schema**.**Types

**{**

**public** class Payload

**{**

**public** bool**?** Success **{** **get;** **}**

**public** IReadOnlyList**<**Error**>?** Errors **{** **get;** **}** **=** **new** List**<**Error**>();**

**public** Payload**(**IReadOnlyList**<**Error**>** errors**,** bool success **=** **true)**

**{**

Success **=** success**;**

Errors **=** errors**;**

**}**

**public** Payload**(**bool success **=** **true)**

**{**

Success **=** success**;**

**}**

**}**

**public** record Error**(**string Field**,** string Info**);**

**}**

Файл Mutation.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Security**.**Claims**;**

**using** HotChocolate**;**

**using** HotChocolate**.**Data**;**

**using** Microsoft**.**AspNetCore**.**Http**;**

**using** Microsoft**.**AspNetCore**.**Identity**;**

**using** server**.**DB**;**

**using** server**.**Models**;**

**using** server**.**Schema**.**Types**;**

**using** Microsoft**.**IdentityModel**.**Tokens**;**

**using** Microsoft**.**AspNetCore**;**

**using** System**.**Text**;**

**using** System**.**Net**;**

**using** System**.**Web**;**

**using** System**.**Text**.**Json**;**

**using** System**.**IO**;**

**using** Microsoft**.**Extensions**.**Options**;**

**using** System**.**IdentityModel**.**Tokens**.**Jwt**;**

**using** server**.**Services**;**

**namespace** server**.**Schema

**{**

**public** class Mutation

**{**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** SignUp**([**Service**]** IIdentityService identity**,** string email**,** string password**)**

**{**

var result **=** **await** identity**.**SignUpAsync**(**email**,** password**);**

var errors **=** **new** List**<**Types**.**Error**>();**

**foreach** **(**var err **in** result**.**Errors**)**

**{**

errors**.**Add**(new** Types**.**Error**(**err**.**Code**,** err**.**Description**));**

**}**

**return** **new** Payload**(**errors**,** result**.**Succeeded**);**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**LoginPayload**>** SignIn**([**Service**]** IIdentityService identity**,** string email**,** string password**)**

**{**

**return** **new** LoginPayload**(await** identity**.**SignInAsync**(**email**,** password**));**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** SignOut**([**Service**]** IIdentityService identity**)**

**{**

**await** identity**.**SignOutAsync**();**

**return** **new** Payload**();**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**LoginPayload**>** RefreshToken**([**Service**]** ITokenService tokenService**,** string refreshToken**)**

**{**

**return** **new** LoginPayload**(await** tokenService**.**UpdateAsync**(**refreshToken**));**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** AddOrder**([**ScopedService**]** ApplicationDbContext context**,**

string mesto**,**

int counts**,**

//int airId,

int flightId**,**

int clientId**,**

int cardId

/\* DateTime dateStart,

DateTime timeStart,

string start,

string target,

string name,

decimal price,

decimal Percent,

string lastName\*/**)**

**{**

var order **=** **new** Order**()**

**{**

Mesto **=** mesto**,**

Counts **=** counts**,**

FlightId **=** flightId**,**

ClientId **=** clientId**,**

CardId **=** cardId

/\* Client = new()

{

LastName = lastName

},

Flight = new()

{

AirId = airId,

DateStart = dateStart,

TimeStart = timeStart,

Route = new()

{

Start = start,

Target = target,

Price = price

},

},

Card = new()

{

Name = name,

Percent = Percent

}\*/

**};**

**await** context**.**Orders**.**AddAsync**(**order**);**

**await** context**.**SaveChangesAsync**();**

**return** **new** Payload**();**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** AddRoute**([**ScopedService**]** ApplicationDbContext context**,** string start**,** string target**,** int time**,** decimal price**)**

**{**

var route **=** **new** Route**()**

**{**

Start **=** start**,**

Target **=** target**,**

Time **=** time**,**

Price **=** price

**};**

**await** context**.**Routes**.**AddAsync**(**route**);**

**await** context**.**SaveChangesAsync**();**

**return** **new** Payload**();**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** AddAir**([**ScopedService**]** ApplicationDbContext context**,**

string bortNumber**,**

string models**,**

int lifeTime**,**

int seats**,**

DateTime dateCreate**,**

bool isActive**,**

int sotrudnikId

**)**

**{**

var air **=** **new** Air**()**

**{**

BortNumber **=** bortNumber**,**

Model **=** models**,**

LifeTime **=** lifeTime**,**

Seats **=** seats**,**

Date\_create **=** dateCreate**,**

IsActive **=** isActive**,**

SotrudnikId **=** sotrudnikId

**};**

**await** context**.**Airs**.**AddAsync**(**air**);**

**await** context**.**SaveChangesAsync**();**

**return** **new** Payload**();**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** AddSotrudnik**([**ScopedService**]** ApplicationDbContext context**,**

string lastName**,**

string name**,**

string surName**,**

string addres**,**

string phone**,**

int stage

**)**

**{**

var sotrudnik **=** **new** Sotrudnik**()**

**{**

Last\_name **=** lastName**,**

Name **=** name**,**

Sur\_name **=** surName**,**

Addres **=** addres**,**

Phone **=** phone**,**

Stage **=** stage

**};**

**await** context**.**Sotrudniks**.**AddAsync**(**sotrudnik**);**

**await** context**.**SaveChangesAsync**();**

**return** **new** Payload**();**

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**public** **async** Task**<**Payload**>** DelRoute**([**ScopedService**]** ApplicationDbContext context**,** int id**,** string start**,** string target**,** int time**,** decimal price**)**

**{**

var route **=** **new** Route**()**

**{**

Id **=** id**,**

Start **=** start**,**

Target **=** target**,**

Time **=** time**,**

Price **=** price

**};**

**await** context**.**Routes**.**AddAsync**(**route**);**

**await** context**.**SaveChangesAsync**();**

**return** **new** Payload**();**

**}**

**}**

**}**

Файл Query.cs

**using** HotChocolate**;**

**using** HotChocolate**.**Data**;**

**using** HotChocolate**.**Types**;**

**using** NetCasbin**;**

**using** server**.**DB**;**

**using** server**.**Models**;**

**using** server**.**Services**;**

**using** server**.**Schema**.**Types**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**namespace** server**.**Schema

**{**

**public** class Query

**{**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Air**>>** GetAirs**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "airs", "read"))

**return** context**.**Airs**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Card**>>** GetCards**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "cards", "read"))

**return** context**.**Cards**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Client**>>** GetClients**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "clients", "read"))

**return** context**.**Clients**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Flight**>>** GetFlights**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "flights", "read"))

**return** context**.**Flights**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Order**>>** GetOrders**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "orders", "read"))

**return** context**.**Orders**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Route**>>** GetRoutes**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "routes", "read"))

**return** context**.**Routes**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UsePaging**]**

**[**UseProjection**]**

**[**UseFiltering**]**

//sorting

**public** **async** Task**<**IQueryable**<**Sotrudnik**>>** GetSotrudniks**([**Service**]** IIdentityService identity**,**

//[Service] Enforcer enforcer,

**[**ScopedService**]** ApplicationDbContext context**)**

**{**

//var user = await identity.GetCurrentUser();

//enforcer.LoadPolicy();

//if (enforcer.Enforce(user.Id, "sotrudniks", "read"))

**return** context**.**Sotrudniks**;**

//throw new System.Exception();

**}**

**[**UseDbContext**(typeof(**ApplicationDbContext**))]**

**[**UseSingleOrDefault**]**

**public** **async** Task**<**User**>** Me**([**Service**]** IIdentityService identity**,** **[**Service**]** Enforcer enforcer**)**

**{**

var user **=** **await** identity**.**GetCurrentUser**();**

enforcer**.**LoadPolicy**();**

**if** **(**enforcer**.**Enforce**(**user**.**Id**,** "users"**,** "read"**))**

**return** user**;**

**throw** **new** System**.**Exception**();**

**}**

**}**

**}**

Файл AddAir.graphql

mutation AddAir(

$bortNumber:String!,

$models:String!

$lifeTime:Int!,

$seats:Int!,

$dateCreate:DateTime!,

$isActive:Boolean!,

$sotrudnikId:Int!

){

addAir(

bortNumber:$bortNumber,

models:$models,

lifeTime:$lifeTime,

seats:$seats,

dateCreate:$dateCreate,

isActive:$isActive,

sotrudnikId:$sotrudnikId

){

success

errors{

field

info

}

}

}

Файл AddOrder.graphql

mutation AddOrder(

$mesto:String!,

$counts:Int!,

$flightId:Int!,

$clientId:Int!,

$cardId:Int!

){

addOrder(

mesto:$mesto,

counts:$counts,

flightId:$flightId,

clientId:$clientId,

cardId:$cardId

){

success

errors{

field

info

}

}

}

Файл AddRoute.graphql

mutation AddRoute(

$start:String!,

$target:String!,

$time:Int!,

$price:Decimal!

){

addRoute(

start:$start,

target:$target,

time:$time,

price:$price

){

success

errors{

field

info

}

}

}

Файл AddSotrudnik.graphql

mutation AddSotrudnik(

$lastName:String!,

$name:String!

$surName:String!,

$addres:String!,

$phone:String!,

$stage:Int!

){

addSotrudnik(

lastName:$lastName,

name:$name,

surName:$surName,

addres:$addres,

phone:$phone,

stage:$stage

){

success

errors{

field

info

}

}

}

Файл GetAirs.graphql

query GetAirs{

airs{

nodes{

id

bortNumber

model

date\_create

lifeTime

isActive

seats

sotrudnik{

last\_name

name

sur\_name

}

}

}

}

Файл GetClients.graphql

query GetClients{

clients{

nodes{

id

lastName

name

surName

phone

addres

passportSeia

passportNumber

}

}

}

Файл GetFlights.graphql

query GetFlights{

flights{

nodes{

id

dateStart

timeStart

isActive

route{

start

target

time

}

air{

seats

}

}

}

}

Файл GetOrders.graphql

query GetOrders{

orders{

nodes{

id

mesto

counts

client{

lastName

name

surName

phone

passportNumber

passportSeia

}

flight{

id

dateStart

timeStart

route{

start

target

time

price

}

}

card{

name

}

}

}

}

Файл GetRoutes.graphql

query GetRoutes{

routes{

nodes{

id

start

target

time

price

}

}

}

Файл GetSotrudniks.graphql

query GetSotrudniks{

sotrudniks{

nodes{

id

last\_name

name

sur\_name

phone

stage

addres

}

}

}

Файл RefreshToken.graphql

mutation RefreshToken($token: String!) {

refreshToken(refreshToken: $token) {

token {

refreshToken

accessToken

}

}

}

Файл SignIn.graphql

mutation SignIn($email:String!, $password:String!) {

signIn(email:$email, password: $password) {

token {

refreshToken

accessToken

}

success

errors {

field

info

}

}

}

Файл schema.graphql

schema {

query: Query

mutation: Mutation

}

type Query {

airs(first: Int after: String last: Int before: String where: AirFilterInput): AirConnection

cards(first: Int after: String last: Int before: String where: CardFilterInput): CardConnection

clients(first: Int after: String last: Int before: String where: ClientFilterInput): ClientConnection

flights(first: Int after: String last: Int before: String where: FlightFilterInput): FlightConnection

orders(first: Int after: String last: Int before: String where: OrderFilterInput): OrderConnection

routes(first: Int after: String last: Int before: String where: RouteFilterInput): RouteConnection

sotrudniks(first: Int after: String last: Int before: String where: SotrudnikFilterInput): SotrudnikConnection

me: User

}

type Mutation {

signUp(email: String! password: String!): Payload!

signIn(email: String! password: String!): LoginPayload!

signOut: Payload!

refreshToken(refreshToken: String!): LoginPayload!

addOrder(mesto: String! counts: Int! flightId: Int! clientId: Int! cardId: Int!): Payload!

addRoute(start: String! target: String! time: Int! price: Decimal!): Payload!

addAir(bortNumber: String! models: String! lifeTime: Int! seats: Int! dateCreate: DateTime! isActive: Boolean! sotrudnikId: Int!): Payload!

addSotrudnik(lastName: String! name: String! surName: String! addres: String! phone: String! stage: Int!): Payload!

delRoute(id: Int! start: String! target: String! time: Int! price: Decimal!): Payload!

}

type Air {

id: Int!

bortNumber: String!

model: String!

date\_create: DateTime!

lifeTime: Int!

seats: Int!

isActive: Boolean!

sotrudnikId: Int!

sotrudnik: Sotrudnik!

flights: [Flight!]!

}

type Card {

id: Int!

name: String!

percent: Decimal!

orders: [Order!]!

}

type Client {

id: Int!

lastName: String!

name: String!

surName: String!

passportSeia: Int!

passportNumber: Int!

addres: String!

phone: String!

orders: [Order!]!

}

type Flight {

id: Int!

dateStart: DateTime!

timeStart: DateTime!

isActive: Boolean!

routeId: Int!

route: Route!

airId: Int!

air: Air!

orders: [Order!]!

}

type Order {

id: Int!

mesto: String!

counts: Int!

flightId: Int!

flight: Flight!

clientId: Int!

client: Client!

cardId: Int!

card: Card!

}

type Route {

id: Int!

start: String!

target: String!

price: Decimal!

time: Int!

flights: [Flight!]!

}

type Sotrudnik {

id: Int!

last\_name: String!

name: String!

sur\_name: String!

addres: String!

phone: String!

stage: Int!

airs: [Air!]!

}

input AirFilterInput {

and: [AirFilterInput!]

or: [AirFilterInput!]

id: ComparableInt32OperationFilterInput

bortNumber: StringOperationFilterInput

model: StringOperationFilterInput

date\_create: ComparableDateTimeOperationFilterInput

lifeTime: ComparableInt32OperationFilterInput

seats: ComparableInt32OperationFilterInput

isActive: BooleanOperationFilterInput

sotrudnikId: ComparableInt32OperationFilterInput

sotrudnik: SotrudnikFilterInput

flights: ListFilterInputTypeOfFlightFilterInput

}

input CardFilterInput {

and: [CardFilterInput!]

or: [CardFilterInput!]

id: ComparableInt32OperationFilterInput

name: StringOperationFilterInput

percent: ComparableDecimalOperationFilterInput

orders: ListFilterInputTypeOfOrderFilterInput

}

input ClientFilterInput {

and: [ClientFilterInput!]

or: [ClientFilterInput!]

id: ComparableInt32OperationFilterInput

lastName: StringOperationFilterInput

name: StringOperationFilterInput

surName: StringOperationFilterInput

passportSeia: ComparableInt32OperationFilterInput

passportNumber: ComparableInt32OperationFilterInput

addres: StringOperationFilterInput

phone: StringOperationFilterInput

orders: ListFilterInputTypeOfOrderFilterInput

}

input FlightFilterInput {

and: [FlightFilterInput!]

or: [FlightFilterInput!]

id: ComparableInt32OperationFilterInput

dateStart: ComparableDateTimeOperationFilterInput

timeStart: ComparableDateTimeOperationFilterInput

isActive: BooleanOperationFilterInput

routeId: ComparableInt32OperationFilterInput

route: RouteFilterInput

airId: ComparableInt32OperationFilterInput

air: AirFilterInput

orders: ListFilterInputTypeOfOrderFilterInput

}

input OrderFilterInput {

and: [OrderFilterInput!]

or: [OrderFilterInput!]

id: ComparableInt32OperationFilterInput

mesto: StringOperationFilterInput

counts: ComparableInt32OperationFilterInput

flightId: ComparableInt32OperationFilterInput

flight: FlightFilterInput

clientId: ComparableInt32OperationFilterInput

client: ClientFilterInput

cardId: ComparableInt32OperationFilterInput

card: CardFilterInput

}

input RouteFilterInput {

and: [RouteFilterInput!]

or: [RouteFilterInput!]

id: ComparableInt32OperationFilterInput

start: StringOperationFilterInput

target: StringOperationFilterInput

price: ComparableDecimalOperationFilterInput

time: ComparableInt32OperationFilterInput

flights: ListFilterInputTypeOfFlightFilterInput

}

input SotrudnikFilterInput {

and: [SotrudnikFilterInput!]

or: [SotrudnikFilterInput!]

id: ComparableInt32OperationFilterInput

last\_name: StringOperationFilterInput

name: StringOperationFilterInput

sur\_name: StringOperationFilterInput

addres: StringOperationFilterInput

phone: StringOperationFilterInput

stage: ComparableInt32OperationFilterInput

airs: ListFilterInputTypeOfAirFilterInput

}

"A connection to a list of items."

type AirConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [AirEdge!]

"A flattened list of the nodes."

nodes: [Air!]

}

"A connection to a list of items."

type CardConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [CardEdge!]

"A flattened list of the nodes."

nodes: [Card!]

}

"A connection to a list of items."

type ClientConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [ClientEdge!]

"A flattened list of the nodes."

nodes: [Client!]

}

"A connection to a list of items."

type FlightConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [FlightEdge!]

"A flattened list of the nodes."

nodes: [Flight!]

}

"A connection to a list of items."

type OrderConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [OrderEdge!]

"A flattened list of the nodes."

nodes: [Order!]

}

"A connection to a list of items."

type RouteConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [RouteEdge!]

"A flattened list of the nodes."

nodes: [Route!]

}

"A connection to a list of items."

type SotrudnikConnection {

"Information to aid in pagination."

pageInfo: PageInfo!

"A list of edges."

edges: [SotrudnikEdge!]

"A flattened list of the nodes."

nodes: [Sotrudnik!]

}

input ComparableInt32OperationFilterInput {

eq: Int

neq: Int

in: [Int!]

nin: [Int!]

gt: Int

ngt: Int

gte: Int

ngte: Int

lt: Int

nlt: Int

lte: Int

nlte: Int

}

input StringOperationFilterInput {

and: [StringOperationFilterInput!]

or: [StringOperationFilterInput!]

eq: String

neq: String

contains: String

ncontains: String

in: [String]

nin: [String]

startsWith: String

nstartsWith: String

endsWith: String

nendsWith: String

}

input ComparableDateTimeOperationFilterInput {

eq: DateTime

neq: DateTime

in: [DateTime!]

nin: [DateTime!]

gt: DateTime

ngt: DateTime

gte: DateTime

ngte: DateTime

lt: DateTime

nlt: DateTime

lte: DateTime

nlte: DateTime

}

input BooleanOperationFilterInput {

eq: Boolean

neq: Boolean

}

input ListFilterInputTypeOfFlightFilterInput {

all: FlightFilterInput

none: FlightFilterInput

some: FlightFilterInput

any: Boolean

}

input ComparableDecimalOperationFilterInput {

eq: Decimal

neq: Decimal

in: [Decimal!]

nin: [Decimal!]

gt: Decimal

ngt: Decimal

gte: Decimal

ngte: Decimal

lt: Decimal

nlt: Decimal

lte: Decimal

nlte: Decimal

}

input ListFilterInputTypeOfOrderFilterInput {

all: OrderFilterInput

none: OrderFilterInput

some: OrderFilterInput

any: Boolean

}

input ListFilterInputTypeOfAirFilterInput {

all: AirFilterInput

none: AirFilterInput

some: AirFilterInput

any: Boolean

}

"Information about pagination in a connection."

type PageInfo {

"Indicates whether more edges exist following the set defined by the clients arguments."

hasNextPage: Boolean!

"Indicates whether more edges exist prior the set defined by the clients arguments."

hasPreviousPage: Boolean!

"When paginating backwards, the cursor to continue."

startCursor: String

"When paginating forwards, the cursor to continue."

endCursor: String

}

"An edge in a connection."

type AirEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Air!

}

"An edge in a connection."

type CardEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Card!

}

"An edge in a connection."

type ClientEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Client!

}

"An edge in a connection."

type FlightEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Flight!

}

"An edge in a connection."

type OrderEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Order!

}

"An edge in a connection."

type RouteEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Route!

}

"An edge in a connection."

type SotrudnikEdge {

"A cursor for use in pagination."

cursor: String!

"The item at the end of the edge."

node: Sotrudnik!

}

type Payload {

success: Boolean

errors: [Error!]

}

type User {

name: String

surname: String

patronymic: String

id: String

userName: String

normalizedUserName: String

email: String

normalizedEmail: String

emailConfirmed: Boolean!

securityStamp: String

concurrencyStamp: String

phoneNumber: String

phoneNumberConfirmed: Boolean!

twoFactorEnabled: Boolean!

lockoutEnd: DateTime

lockoutEnabled: Boolean!

accessFailedCount: Int!

}

type LoginPayload {

token: Token

success: Boolean

errors: [Error!]

}

"The built-in `Decimal` scalar type."

scalar Decimal

"The `DateTime` scalar represents an ISO-8601 compliant date time type."

scalar DateTime

type Token {

id: Int!

refreshToken: String!

accessToken: String!

revoked: Boolean!

createdAt: DateTime!

updatedAt: DateTime!

user\_id: String!

user: User!

}

type Error {

field: String!

info: String!

}

Файл .graphqlrc.json

{

"schema": "schema.graphql",

"documents": "\*\*/\*.graphql",

"extensions": {

"strawberryShake": {

"name": "GqlClient",

"namespace": "client",

"url": "https://localhost:5001/",

"dependencyInjection": true,

"strictSchemaValidation": true,

"hashAlgorithm": "md5",

"useSingleFile": true,

"requestStrategy": "Default",

"outputDirectoryName": "Generated",

"noStore": false,

"emitGeneratedCode": true,

"razorComponents": false,

"records": {

"inputs": false,

"entities": false

},

"transportProfiles": [

{

"default": "Http",

"subscription": "WebSocket"

}

]

}

}

}

Файл FormClient.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormClient **:** Form

**{**

**public** FormClient**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormClient\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormClient formClient **=** **new** FormClient**();**

formClient**.**Close**();**

**}**

**private** **async** void FormClient\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddClient**,** "Добавить клиента"**);**

toolTip**.**SetToolTip**(this.**btnEditClient**,** "Изменить информацию клиента"**);**

toolTip**.**SetToolTip**(this.**btnDeleteClient**,** "Удалить клиента"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться назад"**);**

toolTip**.**SetToolTip**(this.**btnSearch**,** "Поиск клиента"**);**

toolTip**.**SetToolTip**(this.**btnClearSearch**,** "Сброс поиска"**);**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var data **=** **(await** client**.**GetClients**.**ExecuteAsync**().**ConfigureAwait**(true))?.**Data**?.**Clients**;**

**foreach** **(**var clientt **in** data**?.**Nodes **??** Array**.**Empty**<**IGetClients\_Clients\_Nodes**>())**

**{**

dataGridView1**.**Rows**.**Add**(new** **object[]** **{** clientt**.**Id**,**

clientt**.**LastName**,**

clientt**.**Name**,**

clientt**.**SurName**,**

clientt**.**Phone**,**

clientt**.**Addres**,**

clientt**.**PassportSeia**,**

clientt**.**PassportNumber

**});**

**}**

**}**

**}**

**}**

Файл FormControlFlight.cs

**using** client**.**ControlFlight**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms

**namespace** client

**{**

**public** **partial** class FormControlFlight **:** Form

**{**

**public** FormControlFlight**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormControlFlight\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormControlFlight formControlFlight **=** **new** FormControlFlight**();**

formControlFlight**.**Close**();**

**}**

**private** void FormControlFlight\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddControlFlight**,** "Добавить рейс"**);**

toolTip**.**SetToolTip**(this.**btnEditControlFlight**,** "Изменить информацию о рейсе"**);**

toolTip**.**SetToolTip**(this.**btnDeleteControlFlight**,** "Удалить рейс"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться назад"**);**

toolTip**.**SetToolTip**(this.**btnSearch**,** "Поиск клиента"**);**

toolTip**.**SetToolTip**(this.**btnClearSearch**,** "Сброс поиска"**);**

**}**

**private** void btnAddControlFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

FormAddControlFlight formAddControlFlight **=** **new** FormAddControlFlight**();**

formAddControlFlight**.**ShowDialog**();**

**}**

**private** void btnEditControlFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

FormEditControlFlight formEditControlFlight **=** **new** FormEditControlFlight**();**

formEditControlFlight**.**ShowDialog**();**

**}**

**}**

**}**

Файл FormAddControlFlight.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormAddControlFlight **:** Form

**{**

**public** FormAddControlFlight**()**

**{**

InitializeComponent**();**

**}**

**private** void btnAddControlFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddControlFlight**,** "Добавить рейс"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**private** void FormAddControlFlight\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormAddControlFlight formAddControlFlight **=** **new** FormAddControlFlight**();**

formAddControlFlight**.**Close**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormAddControlFlight\_Load**(object** sender**,** EventArgs e**)**

**{**

**}**

**}**

**}**

Файл FormEditControlFlight.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client**.**ControlFlight

**{**

**public** **partial** class FormEditControlFlight **:** Form

**{**

**public** FormEditControlFlight**()**

**{**

InitializeComponent**();**

**}**

**private** void FormEditControlFlight\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnSaveControlFlight**,** "Сохранить изменения"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormEditControlFlight\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormEditControlFlight formEditControlFlight **=** **new** FormEditControlFlight**();**

formEditControlFlight**.**Close**();**

**}**

**private** void btnSaveControlFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

**}**

**}**

**}**

Файл FormFlight.cs

**using** client**.**Flight**;**

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormFlight **:** Form

**{**

**public** FormFlight**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormFlight\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormFlight formFlight **=** **new** FormFlight**();**

formFlight**.**Close**();**

**}**

**private** **async** void FormFlight\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddFlight**,** "Добавить авиалайнер"**);**

toolTip**.**SetToolTip**(this.**btnEditFlight**,** "Изменить информацию о авиалайнере"**);**

toolTip**.**SetToolTip**(this.**btnDeleteFlight**,** "Удалить авиалайнер"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться назад"**);**

toolTip**.**SetToolTip**(this.**btnSearch**,** "Поиск клиента"**);**

toolTip**.**SetToolTip**(this.**btnClearSearch**,** "Сброс поиска"**);**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var data **=** **(await** client**.**GetAirs**.**ExecuteAsync**().**ConfigureAwait**(true))?.**Data**?.**Airs**;**

**foreach** **(**var air **in** data**?.**Nodes **??** Array**.**Empty**<**IGetAirs\_Airs\_Nodes**>())**

**{**

dataGridView1**.**Rows**.**Add**(new** **object[]** **{** air**.**Id**,**

air**.**BortNumber**,**

air**.**Model**,**

air**.**Sotrudnik**.**Last\_name**,**

air**.**Sotrudnik**.**Name**,**

air**.**Sotrudnik**.**Sur\_name**,**

air**.**Date\_create**,**

air**.**LifeTime**,**

air**.**IsActive**,**

air**.**Seats

**});**

**}**

**}**

**private** void btnAddFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

FormAddFlight formAddFlight **=** **new** FormAddFlight**();**

formAddFlight**.**ShowDialog**();**

**}**

**private** void btnEditFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

FormEditFlight formEditFlight **=** **new** FormEditFlight**();**

formEditFlight**.**ShowDialog**();**

**}**

**}**

**}**

Файл FormAddFlight.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client**.**Flight

**{**

**public** **partial** class FormAddFlight **:** Form

**{**

**public** FormAddFlight**()**

**{**

InitializeComponent**();**

**}**

**private** void button2\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormAddFlight\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddFlight**,** "Добавить авиалайнер"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**private** void FormAddFlight\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormAddFlight formAddFlight **=** **new** FormAddFlight**();**

formAddFlight**.**Close**();**

**}**

**private** **async** void btnAddFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

var air **=** **await** AddAir**(**

txtBortNumber**.**Text**,**

txtModel**.**Text**,**

Convert**.**ToInt32**(**txtLifeTime**.**Text**),**

Convert**.**ToInt32**(**txtSeats**.**Text**),**

dateCreate**.**Value**,**

checkBox1**.**Checked**,**

Convert**.**ToInt32**(**cmbSotrudnik**.**Text**)**

**);**

MessageBox**.**Show**(**"Операция произошла"**);**

Close**();**

**}**

**private** **async** Task**<**bool**>** AddAir**(**

string bortNumber**,**

string models**,**

int lifeTime**,**

int seats**,**

DateTime dateCreate**,**

bool isActive**,**

int sotrudnikId**)**

**{**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var res **=** **await** client**.**AddAir**.**ExecuteAsync**(**

bortNumber**,**

models**,**

lifeTime**,**

seats**,**

dateCreate**,**

isActive**,**

sotrudnikId

**).**ConfigureAwait**(true);**

**return** res**.**Errors**.**Count **==** 0**;**

**}**

**}**

**}**

Файл FormEditFlight.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client**.**Flight

**{**

**public** **partial** class FormEditFlight **:** Form

**{**

**public** FormEditFlight**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormEditFlight\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnSaveFlight**,** "Добавить авиалайнер"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**private** void FormEditFlight\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormEditFlight formEditFlight **=** **new** FormEditFlight**();**

formEditFlight**.**Close**();**

**}**

**}**

**}**

Файл FormRoute.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormRoute **:** Form

**{**

**public** FormRoute**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

/\*

this.Hide();

FormAdministration formAdministration = new FormAdministration();

formAdministration.Show();

\*/

**}**

**private** void FormRoute\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormRoute formRoute **=** **new** FormRoute**();**

formRoute**.**Close**();**

//Application.Exit();

**}**

**private** void btnAddRoute\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormAddRoute formAddRoute **=** **new** FormAddRoute**();**

formAddRoute**.**ShowDialog**();**

**}**

**private** void btnEditRoute\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormEditRoute formEditRoute **=** **new** FormEditRoute**();**

formEditRoute**.**ShowDialog**();**

**}**

**private** **async** void FormRoute\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddRoute**,** "Добавить маршрут"**);**

toolTip**.**SetToolTip**(this.**btnEditRoute**,** "Изменить маршрут"**);**

toolTip**.**SetToolTip**(this.**btnDeleteRoute**,** "Удалить маршрут"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться назад"**);**

toolTip**.**SetToolTip**(this.**btnSearch**,** "Поиск маршрута"**);**

toolTip**.**SetToolTip**(this.**btnClearSearch**,** "Сброс поиска"**);**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var data **=** **(await** client**.**GetRoutes**.**ExecuteAsync**().**ConfigureAwait**(true))?.**Data**?.**Routes**;**

**foreach** **(**var route **in** data**?.**Nodes **??** Array**.**Empty**<**IGetRoutes\_Routes\_Nodes**>())**

**{**

dataGridView1**.**Rows**.**Add**(new** **object[]** **{** route**.**Id**,** route**.**Start**,** route**.**Target**,** route**.**Time**,** route**.**Price **});**

**}**

**}**

**}**

**}**

Файл FormAddRoute.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormAddRoute **:** Form

**{**

**public** FormAddRoute**()**

**{**

InitializeComponent**();**

**}**

**private** **async** void btnAddRoute\_Click**(object** sender**,** EventArgs e**)**

**{**

var route **=** **await** AddRoute**(**

txtStart**.**Text**,**

txtTarget**.**Text**,**

Convert**.**ToInt32**(**txtTime**.**Text**),**

Convert**.**ToDecimal**(**txtPrice**.**Text**)**

**);**

MessageBox**.**Show**(**"Операция произошла" /\*+ (await AddRoute(

txtStart.Text,

txtTarget.Text,

Convert.ToInt32(txtTime.Text),

Convert.ToDecimal(txtPrice.Text)).ConfigureAwait(true) ? " успешно" : "не успешно")\*/**);**

Close**();**

**}**

**private** **async** Task**<**bool**>** AddRoute**(**

string start**,**

string target**,**

int time**,**

Decimal price**)**

**{**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var res **=** **await** client**.**AddRoute**.**ExecuteAsync**(**start**,** target**,** time**,** price**).**ConfigureAwait**(true);**

**return** res**.**Errors**.**Count **==** 0**;**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormAddRoute\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormAddRoute formAddRoute **=** **new** FormAddRoute**();**

formAddRoute**.**Close**();**

**}**

**private** void FormAddRoute\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddRoute **,** "Добавить маршрут"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**}**

**}**

Файл FormEditRoute.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormEditRoute **:** Form

**{**

**public** FormEditRoute**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void btnSaveRoute\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormEditRoute\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormEditRoute formEditRoute **=** **new** FormEditRoute**();**

formEditRoute**.**Close**();**

**}**

**private** void FormEditRoute\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnSaveRoute**,** "Сохранить изменения"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**}**

**}**

Файл FormSellTicket.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**namespace** client

**{**

**public** **partial** class FormSellTicket **:** Form

**{**

**public** FormSellTicket**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormSellTicket\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormSellTicket formSellTicket **=** **new** FormSellTicket**();**

formSellTicket**.**Close**();**

**}**

**private** **async** void FormSellTicket\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddSellTicket**,** "Добавить"**);**

toolTip**.**SetToolTip**(this.**btnEditSellTicket**,** "Изменить"**);**

toolTip**.**SetToolTip**(this.**btnDeleteSellTicket**,** "Удалить"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться назад"**);**

toolTip**.**SetToolTip**(this.**btnSearch**,** "Поиск маршрута"**);**

toolTip**.**SetToolTip**(this.**btnClearSearch**,** "Сброс поиска"**);**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var data **=** **(await** client**.**GetOrders**.**ExecuteAsync**().**ConfigureAwait**(true))?.**Data**?.**Orders**;**

**foreach** **(**var order **in** data**?.**Nodes **??** Array**.**Empty**<**IGetOrders\_Orders\_Nodes**>())**

**{**

dataGridView1**.**Rows**.**Add**(new** **object[]** **{** order**.**Id**,**

order**.**Flight**.**Route**.**Start**,**

order**.**Flight**.**Route**.**Target**,**

order**.**Flight**.**Route**.**Time**,**

order**.**Mesto**,**

order**.**Client**.**LastName**,**

order**.**Client**.**Name**,**

order**.**Client**.**SurName**,**

order**.**Client**.**Phone**,**

order**.**Card**.**Name**,**

order**.**Flight**.**Route**.**Price

**});**

**}**

**}**

**private** void btnAddSellTicket\_Click**(object** sender**,** EventArgs e**)**

**{**

**}**

**}**

**}**

Файл FormSotrudnik.cs

**using** client**.**Sotrudnik**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**namespace** client

**{**

**public** **partial** class FormSotrudnik **:** Form

**{**

**public** FormSotrudnik**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormSotrudnik\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormSotrudnik formSotrudnik **=** **new** FormSotrudnik**();**

formSotrudnik**.**Close**();**

**}**

**private** **async** void FormSotrudnik\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddSotrudnik**,** "Добавить сотрудника"**);**

toolTip**.**SetToolTip**(this.**btnEditSotrudnik**,** "Изменить информацию о сотруднике"**);**

toolTip**.**SetToolTip**(this.**btnDeleteSotrudnik**,** "Удалить сотрудника"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться назад"**);**

toolTip**.**SetToolTip**(this.**btnSearch**,** "Поиск клиента"**);**

toolTip**.**SetToolTip**(this.**btnClearSearch**,** "Сброс поиска"**);**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var data **=** **(await** client**.**GetSotrudniks**.**ExecuteAsync**().**ConfigureAwait**(true))?.**Data**?.**Sotrudniks**;**

**foreach** **(**var sotrudnik **in** data**?.**Nodes **??** Array**.**Empty**<**IGetSotrudniks\_Sotrudniks\_Nodes**>())**

**{**

dataGridView1**.**Rows**.**Add**(new** **object[]** **{** sotrudnik**.**Id**,**

sotrudnik**.**Last\_name**,**

sotrudnik**.**Name**,**

sotrudnik**.**Sur\_name**,**

sotrudnik**.**Phone**,**

sotrudnik**.**Stage**,**

sotrudnik**.**Addres

**});**

**}**

**}**

**private** void btnAddSotrudnik\_Click**(object** sender**,** EventArgs e**)**

**{**

FormAddSotrudnik formAddSotrudnik **=** **new** FormAddSotrudnik**();**

formAddSotrudnik**.**ShowDialog**();**

**}**

**private** void btnEditSotrudnik\_Click**(object** sender**,** EventArgs e**)**

**{**

FormEditSotrudnik formEditSotrudnik **=** **new** FormEditSotrudnik**();**

formEditSotrudnik**.**ShowDialog**();**

**}**

**}**

**}**

Файл FormAddSotrudnik.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client**.**Sotrudnik

**{**

**public** **partial** class FormAddSotrudnik **:** Form

**{**

**public** FormAddSotrudnik**()**

**{**

InitializeComponent**();**

**}**

**private** void FormAddSotrudnik\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnAddSotrudnik**,** "Добавить сотрудника"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormAddSotrudnik\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormAddSotrudnik formAddSotrudnik **=** **new** FormAddSotrudnik**();**

formAddSotrudnik**.**Close**();**

**}**

**private** **async** void btnAddSotrudnik\_Click**(object** sender**,** EventArgs e**)**

**{**

var sotrudnik **=** **await** AddSotrudnik**(**

textBox1**.**Text**,**

textBox2**.**Text**,**

textBox3**.**Text**,**

textBox4**.**Text**,**

textBox5**.**Text**,**

Convert**.**ToInt32**(**textBox6**.**Text**)**

**);**

MessageBox**.**Show**(**"Операция произошла"**);**

Close**();**

**}**

**private** **async** Task**<**bool**>** AddSotrudnik**(**

string lastName**,**

string name**,**

string surName**,**

string addres**,**

string phone**,**

int stage**)**

**{**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var res **=** **await** client**.**AddSotrudnik**.**ExecuteAsync**(**

lastName**,**

name**,**

surName**,**

addres**,**

phone**,**

stage

**).**ConfigureAwait**(true);**

**return** res**.**Errors**.**Count **==** 0**;**

**}**

**}**

**}**

Файл FormEditSotrudnik.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client**.**Sotrudnik

**{**

**public** **partial** class FormEditSotrudnik **:** Form

**{**

**public** FormEditSotrudnik**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormEditSotrudnik\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnSaveSotrudnik**,** "Сохранить изменения"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Отмена"**);**

**}**

**private** void FormEditSotrudnik\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormEditSotrudnik formEditSotrudnik **=** **new** FormEditSotrudnik**();**

formEditSotrudnik**.**Close**();**

**}**

**}**

**}**

Файл FormAdministration.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormAdministration **:** Form

**{**

**public** FormAdministration**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Hide**();**

FormMain formMain **=** **new** FormMain**();**

formMain**.**ShowDialog**();**

**}**

**private** void FormAdministration\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

Application**.**Exit**();**

**}**

**private** void btnRoute\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormRoute formRoute **=** **new** FormRoute**();**

formRoute**.**ShowDialog**();**

**}**

**private** void btnFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormFlight formFlight **=** **new** FormFlight**();**

formFlight**.**ShowDialog**();**

**}**

**private** void btnSotrudnik\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormSotrudnik formSotrudnik **=** **new** FormSotrudnik**();**

formSotrudnik**.**ShowDialog**();**

**}**

**private** void btnClient\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormClient formClient **=** **new** FormClient**();**

formClient**.**ShowDialog**();**

**}**

**private** void btnControlFlight\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormControlFlight formControlFlight **=** **new** FormControlFlight**();**

formControlFlight**.**ShowDialog**();**

**}**

**private** void btnSellTicket\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormSellTicket formSellTicket **=** **new** FormSellTicket**();**

formSellTicket**.**ShowDialog**();**

**}**

**private** void FormAdministration\_Load**(object** sender**,** EventArgs e**)**

**{**

toolTip**.**SetToolTip**(this.**btnRoute**,** "Информация о маршрутах"**);**

toolTip**.**SetToolTip**(this.**btnFlight**,** "Информация о авиалайнерах"**);**

toolTip**.**SetToolTip**(this.**btnSotrudnik**,** "Информация о сотрудниках"**);**

toolTip**.**SetToolTip**(this.**btnClient**,** "Информация о клиентах"**);**

toolTip**.**SetToolTip**(this.**btnControlFlight**,** "Информация о рейсах"**);**

toolTip**.**SetToolTip**(this.**btnSellTicket**,** "Информация о заявках"**);**

toolTip**.**SetToolTip**(this.**btnCancel**,** "Вернуться на главное окно"**);**

**}**

**}**

**}**

Файл FormAvtorization.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Threading**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormAvtorization **:** Form

**{**

**private** **readonly** SynchronizationContext**?** syncContext**;**

**public** FormAvtorization**()**

**{**

InitializeComponent**();**

syncContext **=** SynchronizationContext**.**Current**;**

**}**

**private** void Form1\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

Application**.**Exit**();**

**}**

**private** **async** void btnAvtorization\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**txtPasword**.**Text **is** **null** **||** txtLogin**.**Text **is** **null** **||** txtPasword**.**Text**.**Length **==** 0 **||** txtLogin**.**Text**.**Length **==** 0**)**

**return;**

btnAvtorization**.**Enabled **=** **false;**

var res **=** **await** Auth**(**txtLogin**.**Text**,** txtPasword**.**Text**).**ConfigureAwait**(true);**

**if** **(**res **is** not **null)**

**{**

MessageBox**.**Show**(**res**);**

BeginInvoke**((**Action**)(()** **=>** btnAvtorization**.**Enabled **=** **true));**

**return;**

**}**

BeginInvoke**((**Action**)(()** **=>** ShowMenu**()));**

/\*this.Hide();

FormMain formMain = new FormMain();

formMain.Show();\*/

**}**

**private** void label3\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Hide**();**

FormRegistration formRegistration **=** **new** FormRegistration**();**

formRegistration**.**Show**();**

**}**

**private** static **async** Task**<**string**?>** Auth**(**string email**,** string password**)**

**{**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var res **=** **await** client**.**SignIn**.**ExecuteAsync**(**email**,** password**).**ConfigureAwait**(true);**

**if** **(**res**.**Errors**.**Count **>** 0**)**

**{**

**return** res**.**Errors**[**0**].**Message**;**

**}**

Program**.**AccessToken **=** res**?.**Data**?.**SignIn**.**Token**?.**AccessToken **??** ""**;**

Program**.**RefreshToken **=** res**?.**Data**?.**SignIn**.**Token**?.**RefreshToken **??** ""**;**

**return** **null;**

**}**

**private** void ShowMenu**()**

**{**

timer1**.**Start**();**

Hide**();**

FormMain formMain **=** **new** FormMain**();**

formMain**.**ShowDialog**();**

Show**();**

btnAvtorization**.**Enabled **=** **true;**

timer1**.**Stop**();**

**}**

**private** **async** void timer1\_Tick**(object** sender**,** EventArgs e**)**

**{**

var message **=** **await** RefreshToken**().**ConfigureAwait**(true);**

**if** **(**message **is** not **null)**

**{**

BeginInvoke**((**Action**)(()** **=>**

**{**

Array**.**ForEach**(**MdiChildren**,** x **=>** x**.**Close**());**

MessageBox**.**Show**(**message**);**

**}));**

**}**

**}**

**private** static **async** Task**<**string**?>** RefreshToken**()**

**{**

**if** **(**Program**.**RefreshToken **is** **null)**

**{**

**return** "not auth"**;**

**}**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var res **=** **await** client**.**RefreshToken**.**ExecuteAsync**(**Program**.**RefreshToken**).**ConfigureAwait**(false);**

**if** **(**res**.**Errors**.**Count **>** 1**)**

**{**

**return** res**.**Errors**[**0**].**Message**;**

**}**

Program**.**RefreshToken **=** res**.**Data**?.**RefreshToken**?.**Token**?.**RefreshToken **??** ""**;**

Program**.**AccessToken **=** res**.**Data**?.**RefreshToken**?.**Token**?.**AccessToken **??** ""**;**

**return** **null;**

**}**

**}**

**}**

Файл FormMain.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormMain **:** Form

**{**

**public** FormMain**()**

**{**

InitializeComponent**();**

**}**

**private** void formMain\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

Application**.**Exit**();**

**}**

**private** void btnOrderTicket\_Click**(object** sender**,** EventArgs e**)**

**{**

//this.Hide();

FormOrderTicket formOrderTicket **=** **new** FormOrderTicket**();**

formOrderTicket**.**ShowDialog**();**

**}**

**private** void btnAdministration\_Click**(object** sender**,** EventArgs e**)**

**{**

Hide**();**

FormAdministration formAdministration **=** **new** FormAdministration**();**

formAdministration**.**ShowDialog**();**

Show**();**

**}**

**private** **async** void FormMain\_Load**(object** sender**,** EventArgs e**)**

**{**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var data **=** **(await** client**.**GetFlights**.**ExecuteAsync**().**ConfigureAwait**(true))?.**Data**?.**Flights**;**

**foreach** **(**var flight **in** data**?.**Nodes **??** Array**.**Empty**<**IGetFlights\_Flights\_Nodes**>())**

**{**

gridFlightClient**.**Rows**.**Add**(new** **object[]** **{** flight**.**Id**,** flight**.**DateStart**,** flight**.**TimeStart**,** flight**.**IsActive**,** flight**.**Route**.**Start**,** flight**.**Route**.**Target**,**

flight**.**Route**.**Time**,** flight**.**Air**.**Seats**});**

**}**

**}**

**}**

**}**

Файл FormOrderTicket.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormOrderTicket **:** Form

**{**

**public** FormOrderTicket**()**

**{**

InitializeComponent**();**

**}**

**private** void btnCancel\_Click**(object** sender**,** EventArgs e**)**

**{**

Close**();**

**}**

**private** void FormOrderTicket\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

FormOrderTicket formOrderTicket **=** **new** FormOrderTicket**();**

formOrderTicket**.**Close**();**

//Application.Exit();

**}**

**private** **async** void btnBuyTicket\_Click**(object** sender**,** EventArgs e**)**

**{**

var driverId **=** **await** AddOrder**(**

txtMesto**.**Text**,**

Convert**.**ToInt32**(**txtCount**.**Text**),**

Convert**.**ToInt32**(**txtFlightId**.**Text**),**

Convert**.**ToInt32**(**txtClientId**.**Text**),**

Convert**.**ToInt32**(**txtCardId**.**Text**)**

//Convert.ToInt32(txtNumRoute.Text),

//dateTimePicker1.Value,

//cmbStart.Text,

//cmbTarget.Text,

//cmbTypeTicket.Text,

//Convert.ToDecimal(txtPrice.Text)

**);**

MessageBox**.**Show**(**"Операция произошла"**);**

Close**();**

**}**

**private** **async** Task**<**bool**>** AddOrder**(**

string mesto**,**

int counts**,**

int flightId**,**

int clientId**,**

int cardId

//int airId,

//DateTime dateStart,

//DateTime timeStart,

//string start,

//string target,

//string name,

//Decimal price

**)**

**{**

var client **=** Program**.**ServiceProvider**.**GetRequiredService**<**IGqlClient**>();**

var res **=** **await** client**.**AddOrder**.**ExecuteAsync**(**

mesto**,**

counts**,**

flightId**,**

clientId**,**

cardId**).**ConfigureAwait**(true);**

**return** res**.**Errors**.**Count **==** 0**;**

**}**

**}**

**}**

Файл FormRegistration.cs

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

**public** **partial** class FormRegistration **:** Form

**{**

**public** FormRegistration**()**

**{**

InitializeComponent**();**

**}**

**private** void FormRegistration\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

Application**.**Exit**();**

**}**

**private** void btnCreateUser\_Click**(object** sender**,** EventArgs e**)**

**{**

Hide**();**

FormAvtorization formAvtorization **=** **new** FormAvtorization**();**

formAvtorization**.**ShowDialog**();**

**}**

**}**

**}**

Файл Program.cs

**using** Microsoft**.**Extensions**.**DependencyInjection**;**

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Threading**.**Tasks**;**

**using** System**.**Windows**.**Forms**;**

**namespace** client

**{**

static class Program

**{**

**public** static IServiceProvider ServiceProvider **{** **get;** **private** **set;** **}** **=** **default!;**

**public** static string AccessToken **=** ""**;**

**public** static string RefreshToken **=** ""**;**

**private** static void ConfigureServices**()**

**{**

var services **=** **new** ServiceCollection**();**

services**.**

AddGqlClient**()**

**.**ConfigureHttpClient**(**client **=>** **{**

client**.**BaseAddress **=** **new** Uri**(**"https://localhost:5001/"**);**

client**.**DefaultRequestHeaders**.**Authorization **=** **new** System**.**Net**.**Http**.**Headers**.**AuthenticationHeaderValue**(**"Bearer"**,** AccessToken**);**

**});**

ServiceProvider **=** services**.**BuildServiceProvider**();**

**}**

/// <summary>

/// The main entry point for the application.

/// </summary>

**[**STAThread**]**

static void Main**()**

**{**

Application**.**SetHighDpiMode**(**HighDpiMode**.**SystemAware**);**

Application**.**EnableVisualStyles**();**

Application**.**SetCompatibleTextRenderingDefault**(false);**

ConfigureServices**();**

Application**.**Run**(new** FormAvtorization**());**

**}**

**}**

**}**

ПРИЛОЖЕНИЕ Б – ДИСК С разработанной системой