Частное учреждение образования

«Колледж бизнеса и права»

ПРОГРАММНОЕ СРЕДСТВО «ЭМУЛЯТОР ПАРКОВКИ»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Конструирование программ и языки программирования»

КП Т.895017.401

Руководитель проекта ( Е.В. Багласова )

Учащийся ( А. А. Петровский )

2021

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

КП Т.895017.401 ПЗ

Разраб.

*Петровский А.А*

Провер.

Багласова Е.В.

*Программное средство «Эмулятор парковки»*

Лит.

Листов

39

КБП

[Введение 4](#_Toc76420881)

[1 Объектно-ориентированный анализ и проектирование системы 5](#_Toc76420882)

[1.1 Сущность задачи 5](#_Toc76420883)

[1.2 Проектирование модели 5](#_Toc76420884)

[2 Вычислительная система 9](#_Toc76420885)

[2.1 Требования к аппаратным и операционным ресурсам 9](#_Toc76420886)

[2.2 Инструменты разработки 9](#_Toc76420887)

[3 Проектирование задачи 11](#_Toc76420888)

[3.1 Требования к приложению 11](#_Toc76420889)

[3.2 Концептуальный прототип 11](#_Toc76420890)

[3.3 Организация данных 12](#_Toc76420891)

[3.4 Функции и элементы управления 14](#_Toc76420892)

[3.5 Проектирование справочной системы приложения 18](#_Toc76420893)

[4 Описание программного средства 19](#_Toc76420894)

[4.1 Общие сведения 19](#_Toc76420895)

[4.2 Функциональное назначение 19](#_Toc76420896)

[4.4 Выходные данные 20](#_Toc76420897)

[5 Методика испытаний 21](#_Toc76420898)

[5.1 Технические требования 21](#_Toc76420899)

[5.2 Функциональное тестирование 21](#_Toc76420900)

[6 Применение 28](#_Toc76420901)

[6.1 Назначение программы 28](#_Toc76420902)

[6.2 Условия применения 28](#_Toc76420903)

[6.3 Справочная система 28](#_Toc76420904)

[Заключение 30](#_Toc76420905)

[Список информационных источников 31](#_Toc76420906)

[Приложение А 32](#_Toc76420907)

У

# Введение

Целью курсового проекта «Эмулятор парковки» является разработка программного средства «Parking.exe», автоматизирующего процесс учёта машин, использующих парковку и расчёта цены за её использование. Данная программа служит для решения следующих задач: ведение базы данных, содержащей информацию о машинах на парковке, парковочных местах, аккаунтах администраторов парковки, чеках, тарифах; осуществление автоматических расчетов цены использования парковки; формирование чека в word; просмотра занятых свободных и занятых мест; просмотр машин, использующих парковку в данный момент; расчёт прибыли парковки за выбранный администратором отчётный период.

Для достижения цели курсового проекта нужно решить следующие задачи:

* выполнить объектно-ориентированный анализ и проектирование системы, результатом которой будет модель системы;
* определить вычислительную систему, необходимую для создания программного средства;
* по модели выполнить проектирование задачи;
* разработать программное средство;
* описать созданное программное средство;
* выбрать методику испытаний;
* описать процесс тестирования применения;
* описать применение программы.

Решение поставленных задач отражено в пояснительной записке, которая состоит из шести разделов и содержит необходимую и достаточную информацию по использованию данного программного средства.

В первом разделе «Объектно-ориентированный анализ и проектирование системы» рассматривается сущность и актуальность поставленной задачи, описание существующих аналогов, проектирование модели, отображающей функциональную структуру объекта.

Второй раздел «Вычислительная система» имеет описание вычислительной системы, а именно: технические характеристики персонального компьютера (ПК), требования, которые будут предъявляться к персональному компьютеру, описание операционной системы, языка реализации и языка моделирование.

Третий раздел «Проектирование задачи» включает: требования к программному средству, концептуальный прототип, логическую структуру данных в контексте среды разработки, структуру и описание функций пользователя в рамках среды разрабатываемого программного средства, функции и элементы управления, проектирование справочной системы программного средства.

Четвертый раздел «Описание программного средства» отражает общие сведения о программе, функциональное назначение, структуру входных и выходных данных.

В пятом разделе «Методика испытаний» рассматриваются требования к техническим средствам для проведения испытания, требования к характеристикам программы применительно к условиям эксплуатации, требования к информационной и программной совместимости. Также описывается порядок проведения функционального тестирования.

Шестой раздел «Применение» содержит информацию, необходимую в процессе эксплуатации программного средства: его назначение, условия применения, а также справочная система.

В заключении описывается выполнение поставленной задачи, степень соответствия проектных решений задания, причины несоответствия, если таковые имеются.

Графическая часть представлена пятью диаграммами: вариантов использования, классов, последовательности, деятельности и компонентов.

1 Объектно-ориентированный анализ и проектирование системы

# 1.1 Сущность задачи

Приложение «Parking» предназначено для автоматизации контроля и оплаты парковочных мест на парковке, в зависимости от текущего тарифа и длительности проведённого времени. Приложение позволит уменьшить временные затраты на использование и оплату парковки. Приложение заберёт на себя часть обязанностей работников парковки, автоматизирует расчёт прибыли за определённый отчётный период. Обеспечивает автоматическую печать талона и чека в word.

Из всех задач, которые будет решать разрабатываемое программное средство, можно выделить ряд основных:

* ведение базы данных, содержащей информацию о клиентах, парковочных местах, тарифах, чеках, администраторах;
* осуществление контроля занятых мест;
* расчет прибыли за определённый отчётный период;
* формирование талона при въезде на парковку;
* формирование чека при выезде с парковки;
* обеспечение организации интерфейса приложения средствами создания меню, кнопочных форм, панелей инструментов.

Данное программное средство направлено на упрощение работы сотрудника. Такого рода программы упрощают работу, связанную с расчётом цены, учётом занятых мест, редактированием тарифа. Разрабатываемая программа будет автоматически рассчитывать стоимость использования парковки с учётом проведённого времени и действующего тарифа.

Разрабатываемое программное средство будет упрощать работу сотрудника парковки, так как ежедневно осуществляются десятки использований парковки. С помощью данной программы сотрудник будет выполнять только формальную функцию.

Аналога данного программного средства представлено не было.

# 1.2 Проектирование модели

На основании проведенного анализа предметной области и выявленного круга задач, необходимо построить модель, которая будет отображать функциональную структуру объектов программного средства, производимые ими действия и связи между этими действиями.

Главной целью проектирования моделей является отображение функциональной структуры объекта, то есть производимые ими действия и связи между этими действиями. Наиболее распространенным средством моделирования данных являются диаграммы «Сущность-связь», которые предназначены для графического представления моделей данных разрабатываемой программной системы и предлагают некоторый набор стандартных обозначений для определения данных и отношений между ними. С помощью этого вида диаграмм можно описать отдельные компоненты концептуальной модели данных и совокупность взаимосвязей между ними, имеющих важное значение для разрабатываемой системы.

Основными понятиями данной нотации являются понятия сущности и связи. При этом под сущностью понимается произвольное множество реальных или абстрактных объектов, каждый из которых обладает одинаковыми свойствами и характеристиками. В этом случае каждый рассматриваемый объект может являться экземпляром одной и только одной сущности, должен иметь уникальное имя или идентификатор, а также отличаться от других экземпляров данной сущности. Примерами сущностей могут быть: место, чек, талон и так далее.

Связь определяется как отношение или некоторая ассоциация между отдельными сущностями. Примерами связей могут являться родственные отношения типа «отец-сын» или производственные отношения типа «начальник-подчиненный». Другой тип связей задается отношениями «иметь в собственности» или «обладать свойством».

Графическая модель данных строится таким образом, чтобы связи между отдельными сущностями отражали не только семантический характер соответствующего отношения, но и дополнительные аспекты обязательности связей, а также кратность участвующих в данных отношениях экземпляров сущностей [4].

Определим сущности для данного программного средства и построим диаграмму «Сущность-связь». Исследовав предметную область можно выделить следующие сущности, относящиеся к данному дипломному проекту: «Администратор», «Чек», «Талон», «Место», «Тариф».

Диаграмма «Сущность-связь» представлена на рисунке 1.1

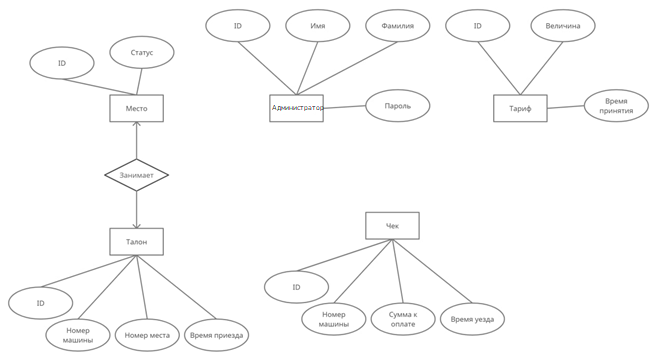


Рисунок 1.1

Для сущности «Место» атрибутами будут являться:

* номер места;
* статус.

Для сущности «Талон» атрибутами будут являться:

* номер талона;
* номер машины;
* номер места;
* время приезда.

Для сущности «Администратор» атрибутами будут являться:

* номер администратора;
* имя;
* фамилия;
* пароль.

Для сущности «Тариф» атрибутами будут являться:

* номер тарифа;
* время принятия;
* величина.

Для сущности «Чек» атрибутами будут являться:

* номер чека;
* номер машины;
* сумма к оплате;
* время уезда.

В качестве инструмента для проектирования модели программного средства будет выбран унифицированный язык моделирования. Это язык для специфицирования, визуализации, конструирования и документирования программных средств [4].

В рамках языка Unified Modeling Language (UML) все представления о модели сложной системы фиксируются в виде специальных графических конструкций – диаграмм. В терминах языка UML определены следующие виды диаграмм: диаграмма вариантов использования, диаграмма классов, диаграммы поведения (диаграмма деятельности), диаграммы взаимодействия (диаграмма последовательности), диаграммы реализации (диаграмма компонентов), диаграмма «Сущность-связь»[4].

Перечень этих диаграмм представляет собой неотъемлемую часть графической нотации языка UML, сам процесс объектно-ориентированного программирования (ООП) неразрывно связан с процессом построения этих диаграмм. Совокупность построенных таким образом диаграмм содержит всю информацию, необходимую для реализации проекта сложной системы.

Диаграмма – граф специального вида, состоящий из вершин в форме геометрических фигур, которые связаны между собой ребрами или дугами.

Суть диаграммы вариантов использования состоит в следующем: проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых вариантов использования.

Варианты использования описывают не только взаимодействия между пользователями и сущностью, но также реакции сущности на получение отдельных сообщений от пользователей и восприятие этих сообщений за пределами сущности. Варианты использования могут включать в себя описание особенностей способов реализации сервиса и различных исключительных ситуаций, таких как корректная обработка ошибок системы. Множество вариантов использования в целом должно определять все возможные стороны ожидаемого поведения системы.

В данной проектируемой системе в качестве актеров выступают клиент и администратор, который служит источником воздействия на моделируемую систему.

К основным функциям разрабатываемого программного средства относятся:

* осуществление ведения базы данных (БД);
* осуществление расчета для формирования цены за использование парковки;
* осуществление расчетов;
* создание талона;
* создание чека.

Диаграмма вариантов использования, отражающая варианты использования приложения для пользователя системы, представлена в графической части на листе 1.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы.

Диаграмма классов представляет собой некоторый граф, вершинами которого являются элементы типа «классификатор», которые связаны различными типами структурных отношений. Следует заметить, что диаграмма классов может также содержать интерфейсы, пакеты, отношения и даже отдельные экземпляры, такие как объекты и связи. Когда говорят о данной диаграмме, имеют в виду статическую структурную модель проектируемой системы. Поэтому диаграмму классов принято считать графическим представлением таких структурных взаимосвязей логической модели системы, которые не зависят или инвариантны от времени.

Диаграмма классов представлена в графической части на листе 2.

При моделировании поведения проектируемой или анализируемой системы возникает необходимость детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Для моделирования процесса выполнения операций в языке UML используются так называемые диаграммы деятельности. Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, переход в следующее состояние срабатывает только при завершении этой операции. Графически диаграмма деятельности представляется в форме графа, вершинами которого являются состояния действия, а дугами – переходы от одного состояния действия к другому.

Основная цель использования диаграмм деятельности – визуализация особенностей реализации операций классов, когда необходимо представить алгоритмы их выполнения. Диаграмма деятельности для функции «Добавление нового администратора» представлена в графической части на листе 3.

Временной аспект поведения имеет существенное значение при моделировании синхронных процессов, описывающих взаимодействия объектов. Именно для этой цели и используются диаграммы последовательности, в которых ключевым моментом является динамика взаимодействия объектов во времени. При этом диаграмма последовательности имеет как бы два измерения: одно – слева направо в виде вертикальных линий, каждая из которых изображает линию жизни отдельного объекта, участвующего во взаимодействии; второе – вертикальная временная ось, направленная сверху вниз, на которой начальному моменту времени соответствует самая верхняя часть диаграммы. Диаграмма последовательности для функции «Добавление нового администратора» представлена в графической части на листе 4.

Диаграмма компонентов описывает объекты реального мира – компоненты программного обеспечения. Эта диаграмма позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами. Диаграмма компонентов представлена в графической части на листе 5.

# 2 Вычислительная система

# 2.1 Требования к аппаратным и операционным ресурсам

Основными минимальными требованиями, выдвигаемыми к аппаратному обеспечению ПК, являются:

* процессор Intel Core i5 – 8250U и выше;
* оперативная память 4096 Мбайт и более;
* свободное место на диске 512 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более;
* монитор;
* мышь, клавиатура;
* принтер.

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

# 2.2 Инструменты разработки

Ниже описаны инструменты разработки, которые будут использоваться для написания программного средства.

Персональный компьютер, на котором будет производится разработка имеет следующую аппаратную конфигурацию:

* процессор Intel Core i5 – 8250U;
* оперативная память 4096 Мбайт;
* видеокарта mx250(2 Гбайт GDDR4);
* клавиатура и мышь.

Для разработки будет использоваться следующее программное обеспечение:

* операционная система Windows 10;
* программная среда разработки Microsoft Visual Studio 2019;
* система управления базами данных Microsoft structured query language (SQL) Server 2019;
* программа Microsoft (MS) Word;
* программа для создания справочной системы Doctor Explain (Dr.Explain).
* язык программирования C#
* технология WPF(Windows Presentation Foundation)

Операционная система – это набор управляющих программ, предназначенных для управления ресурсами вычислительной системы как единого комплекса, другими словами операционная система – это набор программного обеспечения, который обеспечивает работу компьютера. Основными функциями операционной системы являются:

* + управление файловой системой (запись, изменение, копирование файлов, контроль доступа);
  + управление выполнением программ (распределение процессорного времени, загрузка программ с диска в оперативную память, перехват потенциально опасных действий);
  + управление памятью (кэширование, распределение, контроль сохранности данных);
  + диалог с пользователем (чтение команд с клавиатуры, с мыши, вывод информации на экран, на принтер).

При разработки программного средства использовалась операционная система Windows 10, так как на данный момент эта операционная система является самой распространённой операционной системой. В Windows 10 были исправлены практически все недостатки предыдущих операционных систем. Аппаратные требования Windows 10 скромнее, она способна работать даже на маломощных компьютерах и ещё добавлено множество функций, существенно облегчающих работу за компьютерам.

Microsoft Visual Studio 2019 – средство разработки [программного обеспечения](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5), разрабатываемое корпорацией [Microsoft](http://ru.wikipedia.org/wiki/Microsoft) и включающее [язык программирования](http://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и [среду разработки](http://ru.wikipedia.org/wiki/%D0%A1%D1%80%D0%B5%D0%B4%D0%B0_%D1%80%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B8). В то же время Visual Basic 2019 сочетает в себе [процедуры](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и элементы [объектно-ориентированных](http://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) и [компонентно-ориентированных](http://ru.wikipedia.org/w/index.php?title=%D0%9A%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F&action=edit&redlink=1) языков программирования. Среда разработки Visual Basic 2019 включает инструменты для визуального конструирования пользовательского интерфейса.

Microsoft SQL Server — [система управления реляционными базами данных (СУБД)](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D0%A1%D0%A3%D0%91%D0%94), разработанная корпорацией [Microsoft](https://ru.wikipedia.org/wiki/Microsoft). Основной используемый язык запросов — [Transact-SQL](https://ru.wikipedia.org/wiki/Transact-SQL), создан совместно Microsoft и [Sybase](https://ru.wikipedia.org/wiki/Sybase). Transact-SQL является реализацией стандарта [ANSI](https://ru.wikipedia.org/wiki/%D0%90%D0%BC%D0%B5%D1%80%D0%B8%D0%BA%D0%B0%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%BD%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%B8%D0%BD%D1%81%D1%82%D0%B8%D1%82%D1%83%D1%82_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BE%D0%B2)/[ISO](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D0%B6%D0%B4%D1%83%D0%BD%D0%B0%D1%80%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BE%D1%80%D0%B3%D0%B0%D0%BD%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F_%D0%BF%D0%BE_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8) по структурированному языку запросов ([SQL](https://ru.wikipedia.org/wiki/SQL)) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз, данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Doctor Explain (Dr. Explain) – это приложение для создания файлов справки (help-файлов), справочных систем, online руководств пользователя, пособий и технической документации к программному обеспечению и техническим системам

Встроенная технология анализа структуры пользовательского интерфейса позволяет документировать экраны программных приложений практически в автоматическом режиме.

Уникальность заключается в инновационном подходе к созданию пользовательской документации, который значительно ускоряет этот трудоемкий процесс по сравнению с другими инструментами.

Программа способна анализировать пользовательский интерфейс приложений и создавать скриншоты (копии экранов) окон, автоматически расставляя на них пояснительные выноски для элементов интерфейса.

Процесс практически полностью автоматизирован, что позволяет достаточно быстро аннотировать экраны приложений и веб-сайтов для иллюстрирования пользовательской документации на программном обеспечении (ПО).

Microsoft Word – Текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office. Первая версия была написана Ричардом Броди для IBM PC, использующих DOS, в 1983 году. Позднее выпускались версии для Apple Macintosh, SCO UNIX и Microsoft Windows. Текущей версией является Microsoft Office Word 2019 для Windows и macOS, а также веб-версия Word Online, не требующая установки программы на компьютер.

C# – объектно-ориентированный язык программирования. Разработан в 1998–2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

Технология WPF (Windows Presentation Foundation) является часть экосистемы платформы .NET и представляет собой подсистему для построения графических интерфейсов.

Если при создании традиционных приложений на основе WinForms за отрисовку элементов управления и графики отвечали такие части ОС Windows, как User32 и GDI+, то приложения WPF основаны на DirectX. В этом состоит ключевая особенность рендеринга графики в WPF: используя WPF, значительная часть работы по отрисовке графики, как простейших кнопочек, так и сложных 3D-моделей, ложиться на графический процессор на видеокарте, что также позволяет воспользоваться аппаратным ускорением графики.

# 3 Проектирование задачи

# 3.1 Требования к приложению

Разрабатываемое приложение должно иметь понятный и удобный в использовании интерфейс, чтобы взаимодействие между программой и пользователем было максимально упрощено. Для того, чтобы программа выглядела более привлекательно нужно воспользоваться сторонним графическим пользовательским интерфейсом Material Design Theme. Для обучения пользователей необходимо разработать справочную систему, в которой должны быть раскрыты все аспекты работы с программой, возможные трудности, возникшие во время работы и пути их решения.

Кроме того, при разработке форм необходимо соблюдать определённые требования: формы взаимодействия с клиентом не должны содержать лишней информации. Клиент не должен иметь возможности покинуть окно или выйти из приложения. При конструировании форм в необходимых случаях нужно предусмотреть возможность защиты данных от изменения, установить ограничения на некорректный ввод данных. Приложение должно иметь страничную структуру, то есть все изменения интерфейса должны происходить с использованием одного окна.

# 3.2 Концептуальный прототип

Концептуальный прототип представляет собой описание внешнего пользовательского интерфейса – системы меню, диалоговых окон и элементов управления.

Основной интерфейс программного приложения будут представлять формы. Все формы будут содержать стандартные пользовательские элементы управления.

При запуске приложения должно появляется окно с выбором нужного функционала, то есть окна въезда, выезда или перехода в окно администрации. Перед переходом в окно администратора, администратор должен войти в свой аккаунт. Окна администрирования должны иметь возможность: получить информацию о занятых местах; изменение тарифа; удаление и добавления новых аккаунтов администратора; получение информации о прибыли за определённый период.

Окно въезда должно содержать ComboBox содержащий список свободных мест, два TextBox для ввода номера машины и вывода текущего тарифа. Button при нажатии на который производится проверка полей, если поля заполнены верно, то происходит переход на страницу подтверждения.

Окно подтверждения состоит из TextBox, содержащих введённую пользователем информацию и время, когда она была записана.

Окно выбора страницы содержит три кнопки, позволяющие открыть окно въезда, выезда и панель администратора.

Окно выезда содержит TextBox в который вводится номер машины. Button при нажатии на который производится проверка поля, если поле заполнено верно, то происходит переход на страницу чека.

Окно чека содержит в себе TextBox содержащие номер машины, время приезда, время уезда и цену за пользование парковкой.

Окно входа в панель администрации содержит TextBox в которые вводится имя и фамилия администратора и его пароль. Button при нажатии на который производится проверка полей, если поля заполнены верно, то происходит переход на страницу администрации.

Окно администратора содержит ListView в котором находится список машин, стоящих на парковке. Button для перехода на страницу управления, удаления и добавления нового администратора.

Окно управления содержит Calendar для выбора начальной и конечной даты, для отображения прибыли. TextBox для ввода тарифа. Button для расчёта прибыли, изменения тарифа и возвращения на страницу администратора.

# 3.3 Организация данных

Реляционная модель основана на математическом понятии отношения, представлением которого является таблица. В реляционной модели отношения используются для хранения информации об объектах, представленных в базе данных. Отношение имеет вид двухмерной таблицы, в которой строки соответствуют записям, а столбцы – атрибутам. Каждая запись должна однозначно характеризоваться в таблице. Для этого используют первичные и вторичные ключи. Достоинством реляционной модели является простота и удобство физической реализации.

Реляционная модель базы данных подразумевает нормализацию всех таблиц данных. Нормализация – это формальный метод анализа отношений на основе их первичного ключа и функциональных зависимостей, существующих между их атрибутами.

База данных соответствует реляционной модели данных, где каждый выделенный в ходе проектировании сущности соответствует таблица.

Структура базы данных разрабатываемого программного средства включает пять таблиц.

Таблица «Администратор» хранит имя, фамилию и пароль, структура приведена в таблице 3.1.

Таблица 3.1 – Структура таблицы «Администратор»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,  байт | Описание |
| id | int | 4 | Идентификатор администратора |
| name | nchar | 30 | Имя администратора |
| surname | nchar | 30 | Фамилия администратора |
| pass | nchar | 36 | Пароль администратора |

В таблице «Место» хранится информация о статусе и номер места, структура приведена в таблице 3.2.

Таблица 3.2 – Структура таблицы «Место»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| id | int | 4 | Идентификатор места |
| status | nchar | 18 | Статус места |

Таблица «Чек» хранит информацию о времени выезда, стоимости, номере машины, структура приведена в таблице 3.3

Таблица 3.3 – Структура таблицы «Чек»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер, байт | Описание |
| id | int | 4 | Идентификатор чека |
| departure | datetime | 8 | Время уезда |
| pay | money | 8 | Стоимость |
| car | nchar | 20 | Номер машины |

В таблице «Талон» представлена информация о номере машины, номере места, времени въезда, структура приведена в таблице 3.4.

Таблица 3.4 – Структура таблицы «Талон»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,байт | Описание |
| id | int | 4 | Идентификатор талона |
| car | nchar | 20 | Номер машины |
| place | int | 4 | Идентификатор места |
| arrival | datetime | 8 | Время приезда |

Таблица «Тариф» хранит данные о времени изменения, величине тарифа, структура приведена в таблице 3.5.

Таблица 3.5 – Структура таблицы «Тариф»

|  |  |  |  |
| --- | --- | --- | --- |
| Имя поля | Тип данных | Размер,байт | Описание |
| id | int | 4 | Идентификатор тарифа |
| ddate | datetime | 8 | Дата изменения |
| pay | money | 8 | Величина тарифа |

Логическая структура базы данных представлена схемой данных на рисунке 3.1.

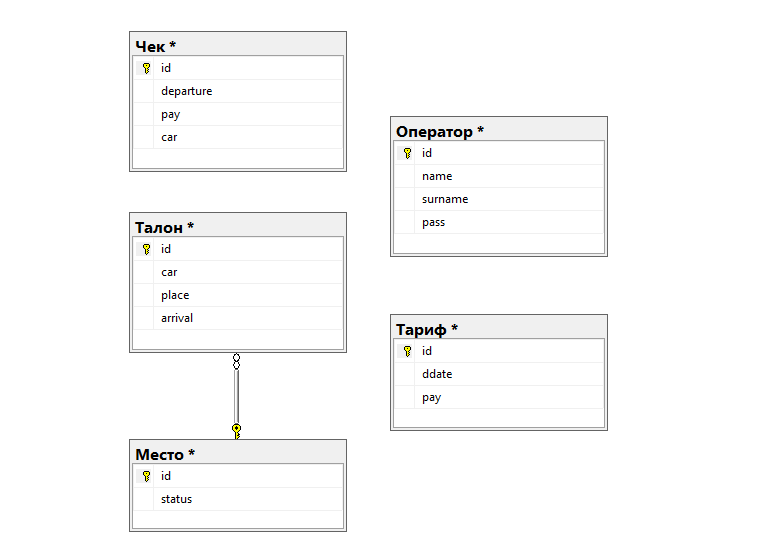


Рисунок 3.1

# 3.4 Функции и элементы управления

Обязательными функциями программы являются занесение машины клиента в базу данных, расчёт стоимости парковки, расчёт прибыли за отчётный период.

Функции, которые выполняет программа – печать талона о приезде, печать чека, занесение номера машины в базу данных, запись времени приезда и уезда, расчёт стоимости в зависимости от времени и действующего тарифа, добавление и удаление администраторов, просмотр прибыли за отчётный период.

Для успешного выполнения вышеперечисленных функций программу необходимо запустить на персональном компьютере (ПК).

Функция «Занесение машины в базу данных» закреплена за элементом управления типа Button. При нажатии на кнопку «Добавить» вызывается метод Resume класса Entrance. Функция требует ввод номера машины, выбор свободного места, если оно присутствует. Код функции представлен ниже:

private void Resume(object sender, RoutedEventArgs e)

{

string num = NumberInput.Text.ToUpper();

bool b = Regex.IsMatch(num, @"^\d{4}\s[A-Z]{2}-+[1-7]$");

bool stations = Regex.IsMatch(num, @"^\d{4}\s[A-Z]{2}-+[0]$");

if (stations)

{

MessageBox.Show("Гос номера не обслуживаем!");

Application.Current.Shutdown();

return;

}

//Проверка правильности ввода значений

if (!b)

{

NumberInput.Foreground = Brushes.Red;

NumberInput.ToolTip = "Неверный формат";

}

else

{

NumberInput.Foreground = Brushes.Black;

NumberInput.ToolTip = "";

}

if (b && ComboBoxView.SelectedIndex != -1)

{

DataTable tableReport = SQLbase.Select($"select \* from Report where car = '{num}'");

if(tableReport.Rows.Count >= 1)

{

NumberInput.Foreground = Brushes.Red;

NumberInput.ToolTip = "Машина уже присутствует.";

return;

}

else

{

NumberInput.Foreground = Brushes.Black;

NumberInput.ToolTip = "";

}

int index = ComboBoxView.SelectedIndex + 1;

string s = ComboBoxView.Text;

string time = DateTime.Now.ToString();

SQLbase.Insert($"INSERT INTO Report(car, place, arrival) values('{num}', {s}, '{time}');");

SQLbase.Insert($"UPDATE Place SET status = 'Занято' where id = {s}");

//Переход на другую страницу с параметром

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

Confirmation nextPage = new Confirmation(num, s, time);

nav.Navigate(nextPage);

}

}

Функция «Расчёт стоимости парковки» закреплена за методом Resume класса Payment. Функция требует ввод номера машины, находящейся на парковке. Код функции представлен ниже:

private void Resume(object sender, RoutedEventArgs e)

{

string num = NumberInput.Text.ToUpper();

if (num.Length == 0)

{

NumberInput.Foreground = Brushes.Red;

NumberInput.ToolTip = "Заполните поле";

}

else

{

NumberInput.Foreground = Brushes.Black;

NumberInput.ToolTip = "";

}

DataTable table = SQLbase.Select($"select \* from Report where car = '{num}'");

if (table.Rows.Count >= 1)

{

string dateInput = table.Rows[0][3].ToString();

DateTime dateIn = DateTime.Parse(dateInput);

DateTime dateExit = DateTime.Now;

DataTable t = SQLbase.Select($"select pay, ddate from Rate order by ddate desc");

decimal rate = Convert.ToDecimal(t.Rows[0][0]);

if (t.Rows.Count != 0)

{

int i = 0;

try

{

while (DateTime.Parse(t.Rows[i][1].ToString()) > DateTime.Parse(table.Rows[0][3].ToString()))

{

rate = Convert.ToDecimal(t.Rows[i][0]);

i++;

}

}

catch

{

rate = 0;

}

}

int hour = Convert.ToInt32((dateExit - dateIn).TotalHours) - 1;

if(hour < 0)

{

hour = 0;

}

decimal sum = hour \* rate;

SQLbase.Insert($"insert into Payment(departure, pay, car) values('{dateExit.ToString()}', {sum.ToString().Replace(",",".")}, '{num}');");

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

Receipt nextPage = new Receipt(num);

nav.Navigate(nextPage);

}

else

{

NumberInput.Foreground = Brushes.Red;

NumberInput.ToolTip = "Неверный номер машины";

}

}

Функция «Расчёт прибыли за отчетный период» закреплена за методом Calculate класса Price. Требуется выбрать начальную и конечную дату отчетного периода, а после вызвать функцию нажатием кнопки «рассчитать». Код функции представлен ниже:

private void Calculate(object sender, RoutedEventArgs e)

{

if(startDate.SelectedDate == null || endDate.SelectedDate == null)

{

editButton.Foreground = Brushes.Red;

editButton.ToolTip = "Выберите обе даты";

return;

}

else

{

editButton.Foreground = Brushes.AliceBlue;

editButton.ToolTip = "";

}

DateTime start = startDate.SelectedDate.Value;

DateTime end = endDate.SelectedDate.Value;

using (DataTable Place = SQLbase.Select($"select sum(pay) from Payment where departure >= '{start.Day}-{start.Month}-{start.Year}' and departure <= '{end.Day}-{end.Month}-{end.Year}'"))

{

string str;

if (Place.Rows[0][0].ToString() == null || Place.Rows[0][0].ToString() == "" || Place.Rows[0][0].ToString() == " ")

{

str = "0";

}

else

{

str = String.Format("{0:C}", Place.Rows[0][0]);

}

answerOutput.Content = $"{str}";

}

}

Текст программы представлен в приложении А.

# 3.5 Проектирование справочной системы приложения

Для работы с приложением начинающего пользователя необходимо обеспечить качественной справочной системой, в которой должны быть приведены методы и приемы работы с приложением, включающие данные о том, что произойдет после нажатия на определенную кнопку или при выборе пункта меню.

Справочная система представляет собой приложение «Parking.chm», поставляемое вместе с программой.

Справочная система необходима для ознакомления с программой. В ней должна присутствовать информация, которая может пригодиться пользователю: о правилах пользования приложением и о его возможностях.

Система справки данного программного средства будет содержать следующие разделы:

* «Начальное окно»;
* «Окно въезда»;
* «Окно выезда»;
* «Окно администратора»;
* «Окно управления».

Справочная система будет создана с помощью программного средства Dr.Explain.

# 4 Описание программного средства

# 4.1 Общие сведения

Приложение «Parking» предназначено для автоматизации контроля и оплаты парковочных мест на парковке, в зависимости от текущего тарифа и длительности проведённого времени. Приложение позволит уменьшить временные затраты на использование и оплату парковки. Приложение заберёт на себя часть обязанностей работников парковки, автоматизирует расчёт прибыли за определённый отчётный период. Обеспечивает автоматическую печать талона и чека в word.

Данное программное средство было разработано при использовании персонального компьютера со следующей конфигурацией:

* процессор Intel Core i5 – 8250U;
* оперативная память 4096 Мбайт;
* твёрдотельный накопитель на 256 Гбайт;
* видеокарта mx250 на 2048 Мбайт;

Программное средство создано в среде разработки Visual Studio 2019 на языке программирования C# в операционной системе Windows 10. Программное средство может работать в средах операционных систем семейства Microsoft Windows начиная с Windows 7. Программа не требовательна к системным ресурсам, также проста в использовании и не требует специальных навыков при работе. Для работы данного программного средства требуется предварительная установка и настройка следующих программных продуктов:

* система управления базами данных Microsoft SQL Server 2019;
* программная платформа Microsoft .Net Framework 4.7.2.

Инсталляция программного средства не требуется, достаточно только скопировать готовый проект на ПК и запустить. Название программного средства – Parking.exe. Для входа в меню администратора необходимо указать логин и пароль. Логин: Admin; пароль: Admin.

# 4.2 Функциональное назначение

Программное средство было разработано с целью уменьшения участия работников в процессе оформления клиентов. Уменьшение временных затрат при составлении отчетов, быстрое использовать данных, хранящиеся в системе для учёта прибыли за отчётный период.

Основными задачами приложения являются автоматизированная печать талона и чека, расчёт цена за использование парковки.

Программа использует стандартные элементы управления, такие как кнопки, меню, списки, поля ввода, что обеспечивает единство интерфейса системы и программного средства.

В работе программного средства предусмотрены некоторые ситуации, которые должны предупреждать пользователя, чтобы он выполнял все необходимые требования по эксплуатации программы. Для этого существуют сообщения системы, например, проверка на корректность ввода номера автомобиля, правильность ввода данных администратора при входе и регистрации.

Таким образом, программа может применяться в реальных условиях, представляя собой достаточно удобный помощник.

4.3 Входные данные

Входными являются данные о тарифах, местах, чеках, талонах, администраторах, которые необходимы для взаимодействия с программой

Входными данными при добавлении талона являются:

* номер машины;
* выбранное место.

Входными данными при добавлении администратора являются:

* имя;
* фамилия;
* пароль.

Входными данными при добавлении чека является номер машины.

Входными данными при добавлении тарифа является величина тарифа:

# 4.4 Выходные данные

Выходные данные – это сформированная стоимость за использование парковки с учётом тарифа и проведённого времени, за выбранный отчётный период.

Выходными данными программы являются:

* стоимость использования парковки;
* общая прибыль за определённый период;
* талон с информацией о приезде;
* чек с суммой оплаты и временем приезда и выезда.

# 5 Методика испытаний

# 5.1 Технические требования

Минимальными требованиями для оптимальной работы программного средства является персональный компьютер со следующими характеристиками:

* процессор Intel Core i5 – 8250U и выше;
* оперативная память 4096 Мбайт и более;
* свободное место на диске 512 Мбайт;
* интегрированная видеокарта на 512 Мбайт и более.

Компьютер должен работать под управлением операционной системы, начиная с Windows 7 и выше. Наиболее удобной операционной системой для проведения испытаний является Windows 10, так как она ориентированна на максимальное использование всех возможностей ПК, сетевых ресурсов и обеспечение комфортных условий работы.

# 5.2 Функциональное тестирование

Функциональное тестирование – это тестирование ПО в целях проверки реализуемости функциональных требований, то есть способности ПО в определенных условиях решать задачи, нужные пользователем. Функциональные требования определяют, что именно делает ПО, какие задачи оно решает. При функциональном тестировании осуществляется проверка каждого пункта меню, каждой операции с целью проверки выполнения всех функций, определенных на этапе проектирования задачи. Функциональное тестирование должно гарантировать работу всех элементов управления в автономном режиме.

Тестирование было реализовано при помощи тест-кейсов, которые представлены в таблицах 5.1-5.7. Тестирование проводилось на уровне разработчика программного средства. Допускаются погрешность временных показателей связанных с процессом тестирования.

Данное тестирование проводится для выявления неполадок и недочетов программы на этапе ее сдачи в эксплуатацию.

Функциональное тестирование предполагает проверку выполнения всех определенных на этапе проектирования функций.

Таблица 5.1 – Тест-кейс функции оформление въезда машины

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 1 | Оформление въезда | 1. Запустить программу  2. В главном меню, нажать кнопку «Окно въезда»  3. Заполнить форму  4. Нажать на кнопку «Продолжить»  5. Проверить правильность значений на странице «Подтверждение», | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.2  Окно подтверждения представлено на рисунке 5.3  Машина внесена в базу данных. Результат представлен рисунке 5.4 |

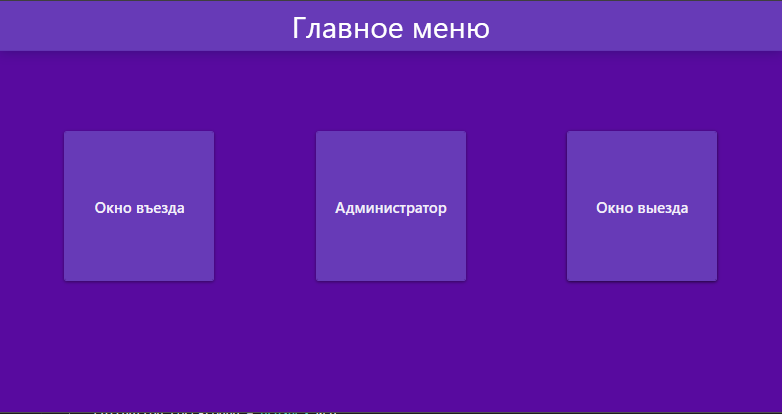


Рисунок 5.1 – Главное меню

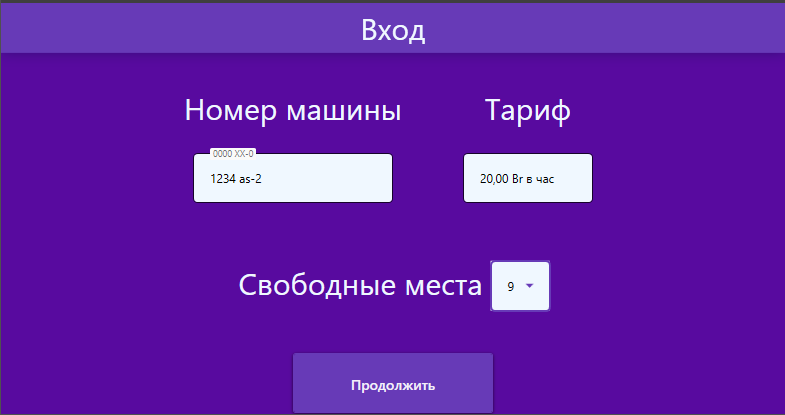


Рисунок 5.2 – Окно въезда

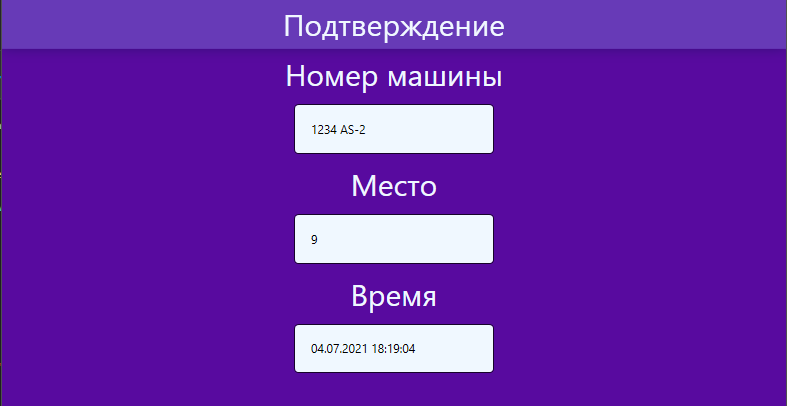


Рисунок 5.3 – Окно подтверждения

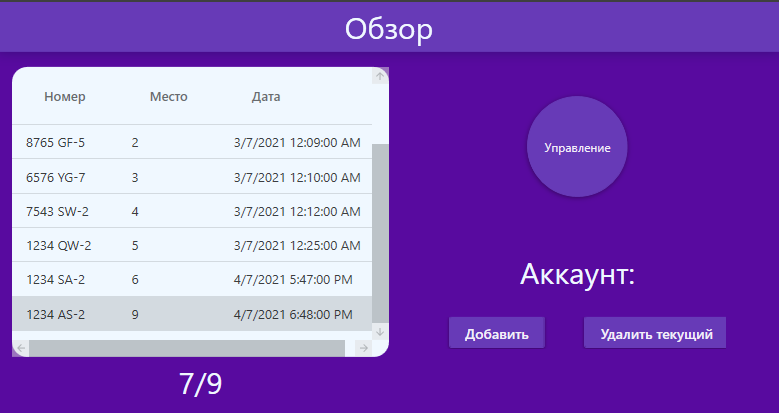


Рисунок 5.4 – Окно обзора

Таблица 5.2 – Тест-кейс функции оформление выезда машины

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 2 | Оформление выезда | 1. Запустить программу  2. В главном меню нажать кнопку «Окно выезда»  3. Заполнить форму  4. Нажать на кнопку «Продолжить»  5. Проверить правильность значений на странице «Оплата» | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.5  Окно «Оплата» представлено на рисунке 5.6  Машина удалена из базы данных. Результат представлен рисунке 5.7 |

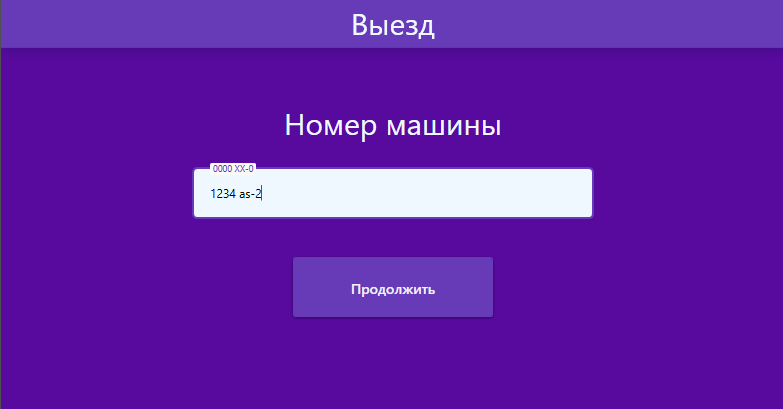


Рисунок 5.5 – Окно выезда

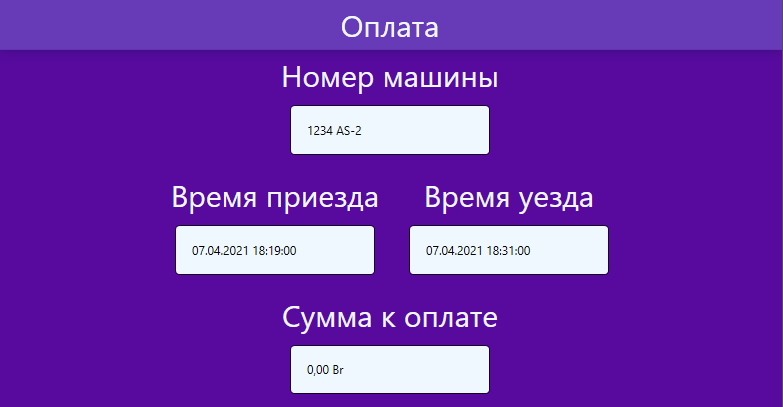


Рисунок 5.6 – Окно оплаты

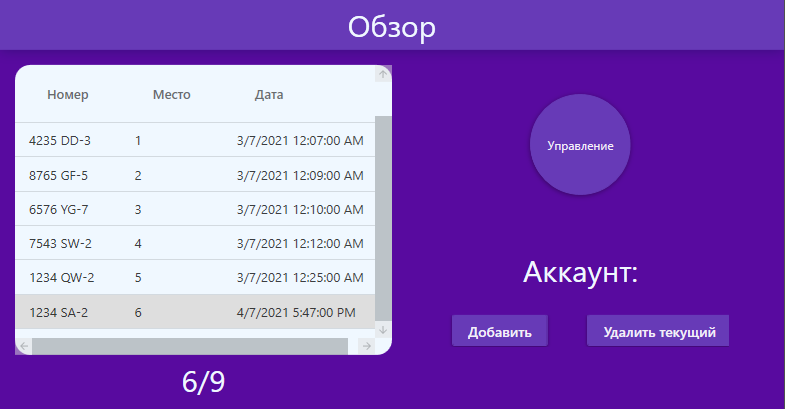


Рисунок 5.7 – Окно обзора

Таблица 5.3 – Тест-кейс функции входа в панель администратора

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 3 | Вход в панель администратора | 1. Запустить программу  2. В главном меню нажать кнопку «Администратор»  3. Заполнить форму  4. Нажать на кнопку «Продолжить» | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.8  Результат входа представлен на рисунке 5.7 |

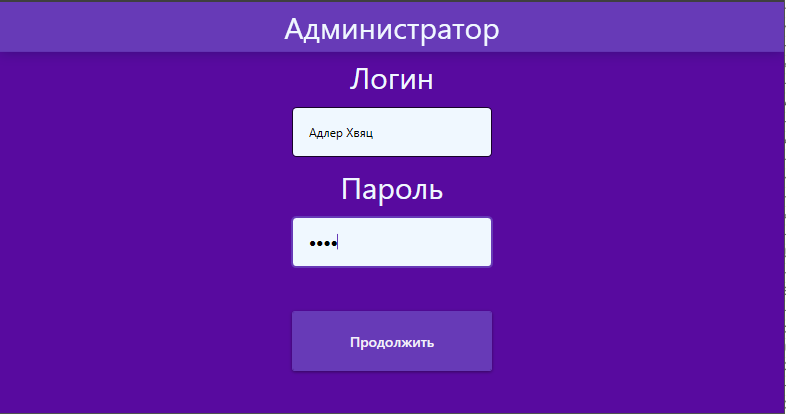


Рисунок 5.8 – Окно входа

Таблица 5.4 – Тест-кейс функции добавления нового администратора

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 4 | Добавление администратора | 1. Запустить программу  2. В главном меню нажать кнопку «Администратор»  3. Заполнить форму  4. Нажать на кнопку «Продолжить»  5. Нажать на кнопку «Добавить»  6. Заполнить форму регистрации | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.8  Форма регистрации представлена на рисунке 5.9  Новый администратор добавлен. Результат подтверждения добавления администратора представлено на рисунке 5.10 |

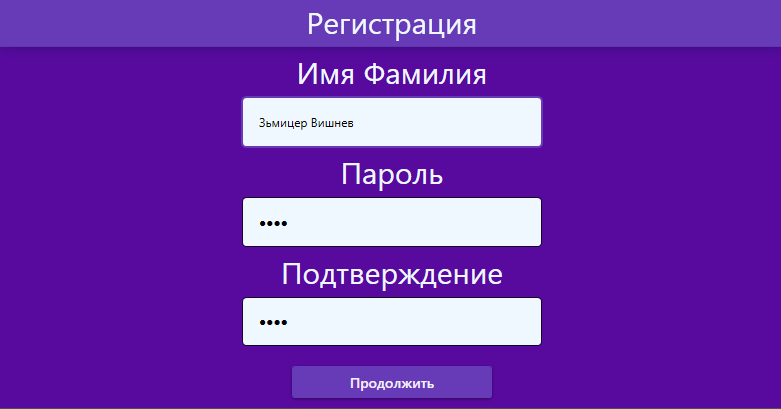


Рисунок 5.9 – Окно регистрации

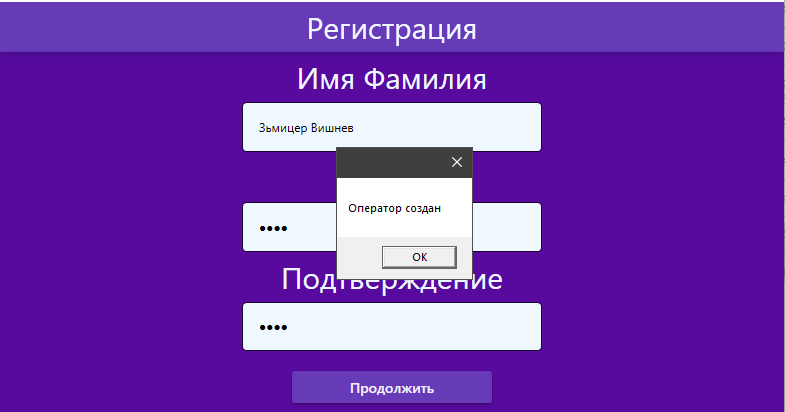


Рисунок 5.10 – Подтверждение добавления администратора

Таблица 5.5 – Тест-кейс функции удаления администратора

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 5 | Удаление администратора | 1. Запустить программу  2. В главном меню нажать кнопку «Администратор»  3. Заполнить форму  4. Нажать на кнопку «Продолжить»  5. Нажать на кнопку «Удалить текущий» | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.8  Администратор удалён. Результат подтверждения удаления администратора представлен на рисунке 5.11 |

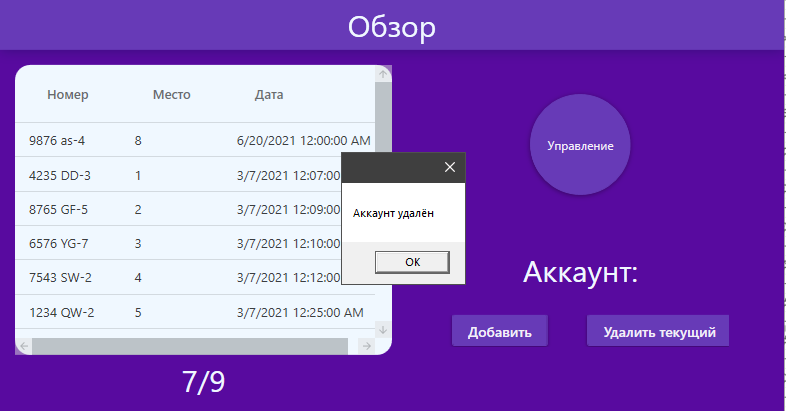


Рисунок 5.11 – Подтверждение удаления

Таблица 5.6 – Тест-кейс функции изменения тарифа

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 6 | Изменение тарифа | 1. Запустить программу  2. В главном меню нажать кнопку «Администратор»  3. Заполнить форму  4. Нажать на кнопку «Продолжить»  5. Нажать на кнопку «Управление» на странице обзора  6. Заполнить форму изменения тарифа  7. Нажать на кнопку «Применить» в окне управления | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.8  Окно обзора представлено на рисунке 5.7  Окно управления представлено на рисунке 5.12  Тариф изменён. Результат изменения тарифа представлен на рисунке 5.13 |

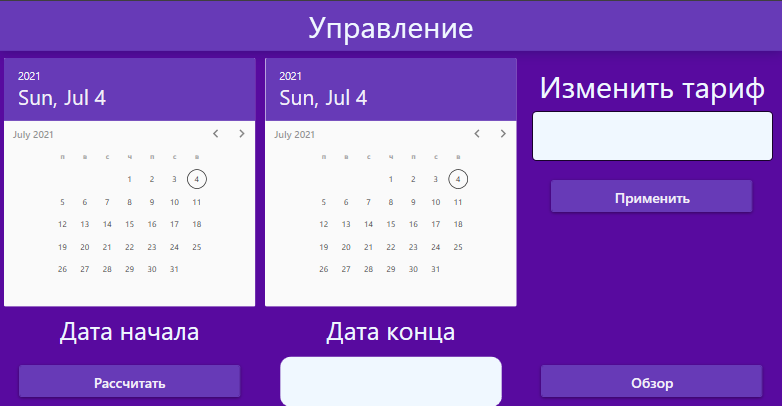


Рисунок 5.12 – Окно управления

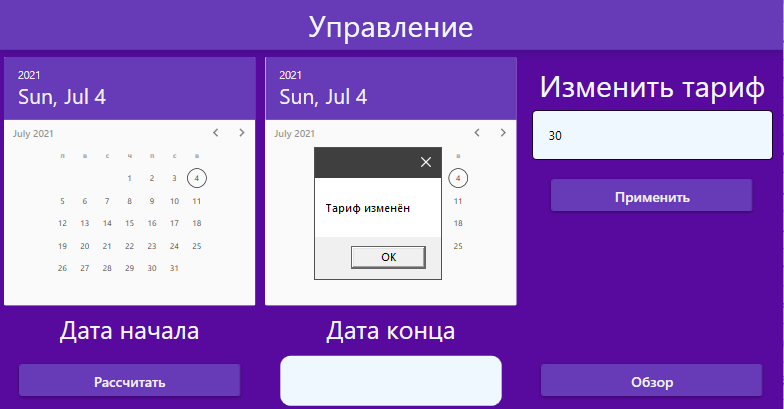


Рисунок 5.13 – Подтверждение изменения тарифа

Таблица 5.7 – Тест-кейс функции расчёта прибыли за выбранный период

|  |  |  |  |
| --- | --- | --- | --- |
| № | Функция | Шаги выполнения | Результат |
| 7 | Расчёт прибыли за выбранный период | 1. Запустить программу  2. В главном меню нажать кнопку «Администратор»  3. Заполнить форму  4. Нажать на кнопку «Продолжить»  5. Нажать на кнопку «Управление» в окне обзора  6. Выбрать на календарях дату начала и конца отчётного периода в окне управления  7. Нажать на кнопку «Рассчитать» | Главное меню представлено на рисунке 5.1  Форма представлена на рисунке 5.8  Окно обзора представлено на рисунке 5.7  Окно управления представлено на рисунке 5.12  Была рассчитана прибыль за выбранный отчётный период и выведена на экран. Результат расчёта представлен на рисунке 5.14 |

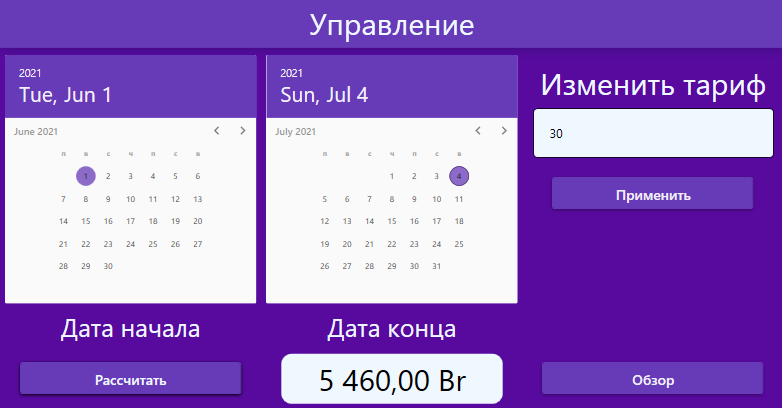


Рисунок 5.14 – Прибыль за выбранный отчётный период

# 6 Применение

# 6.1 Назначение программы

Программа для автоматизации оформления клиентов парковки и расчёта цены.

Основной целью программного средства является уменьшение участия работников в процессе оформления клиентов. Уменьшение временных затрат при составлении отчетов, быстрое использовать данных, хранящиеся в системе для учёта прибыли за отчётный период.

База данных проекта содержит необходимые таблицы для реализации поставленных задач.

Предоставлена возможность добавления новых администраторов парковки и их удаление. Изменение тарифа, за использование парковки, печать чека и талона для клиента.

# 6.2 Условия применения

Для применения данного программного средства необходимы следующие технические требования:

* процессор Intel Core i5 – 8250U;
* оперативная память 4096 Мбайт;
* твёрдотельный накопитель на 256 Гбайт;
* видеокарта mx250 на 2048 ГМбайт;
* операционная система Windows 7 и выше;
* Framework v4.7.5;
* рекомендуется монитор типа VGA или с лучшей разрешающей способностью;
* клавиатура;
* мышь;
* принтер.

# 6.3 Справочная система

Справочная система программного средства представляет собой отдельный файл «Help.chm» с описанием основных функций программы в формате \*.chm. В справочной системе даны ответы на типичные вопросы, возникающие при работе с приложением, что, несомненно, должно помочь при освоении программного средства.

Система справки данного программного средства будет содержать следующие разделы:

* «Начальное окно»;
* «Окно въезда»;
* «Окно выезда»;
* «Окно администратора»;
* «Окно управления».

Структура справочной системы представлена на рисунке 6.1.

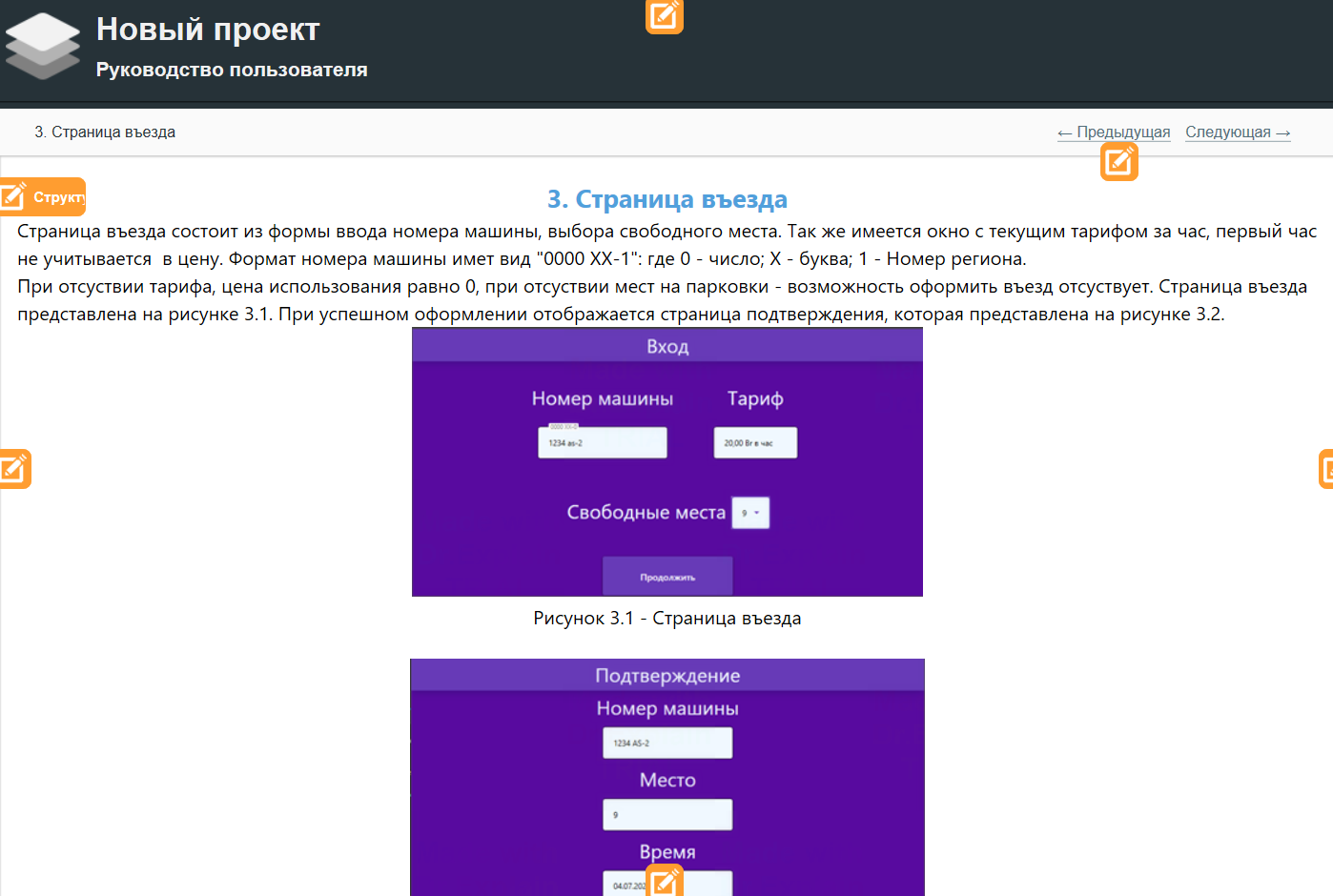


Рисунок 6.1

# Заключение

В рамках курсового проектирования на тему программное средство «Эмулятор парковки» было разработано программное средство, автоматизирующее оформление клиента парковки и расчёта цены.

Для достижения цели были решены следующие задачи:

* выполнен объектно-ориентированный анализ и проектирование системы, результатом которой будет модель системы;
* определена вычислительная система, необходимая для создания программного средства;
* по модели выполнено проектирование задачи;
* разработано программное средство;
* описано созданное программное средство;
* выбрана методика испытаний;
* описан процесс тестирования применения;
* приведены примеры области.

Программа реализована в полном объеме и в соответствии с заданными требованиями, полностью отлажена и протестирована. Поставленные задачи выполнены.

Разработанная база позволяет получить всю необходимую информацию о прибыли за отчётный период, осуществить смену тарифа, добавление и удаление администраторов, сформировать талон и чек, содержащие необходимую клиенту информацию.

При разработке приложения наибольшее внимание уделялось максимальному созданию лаконичного интерфейса.

Проект был разработан в среде Microsoft Visual Studio 2019 на языке C# при разработке базы данных Microsoft SQL Server 2019.

# Список информационных источников

1. Багласова, Т.Г. Методические указания по оформлению курсовых и дипломных проектов / Т.Г. Багласова, К.О. Якимович. – Минск : КБП, 2013. – 29 c.
2. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. / А.М. Вендров. – 2-е изд., перераб. и доп.– М. : Финансы и статистика, 2006. — 544 с.
3. Макфарланд, Д. Большая книга С# / Д. Макфарланд. – 3-е изд. – СПб. : Питер, 2014. — 608 с.
4. Мацяшек. Анализ и проектирование информационных систем с помощью UML 2.0 / Мацяшек, А. М. Лешек: – М. – ООО “И.Д. Вильямс”, 2008. – 816 с.
5. Орлов С. А. Технологии разработки программного обеспечения: Учебник для вузов. 4-е изд. / С. А. Орлов, Б. Я. Цилькер – СПб. : Питер, 2012. — 608 с.
6. Просницкий А. Самоучитель «Управление проектами в Microsoft Project 2010» / А. Просницкий. – Київ, Казань, Красноярск, Рига, Санкт-Петербург, Львів, Чернівці, Івано-Франківськ, 2011.
7. Рудаков А.В., Технология разработки программных продуктов. Практикум: учеб. пособие для студ. учреждений сред. проф. образования / А.В. Рудаков, Г.Н. Федорова. 4-е изд., стер. – М. : Издательский центр «Академия»; 2014. — 192 с.
8. Тепляков С. Паттерны проектирования на платформе .NET. / С. Тепляков – СПб. : Питер, 2018. – 317 с.
9. Экономика предприятия. Практикум / Э. В. Крум [и др.] ; под ред. Э. В. Крум. – Минск : Издательство Гревцова, 2009. – 355 с.
10. Язык программирования C#. Классика Computers Science / А. Хейлсберг [и др]. – 4-е изд. – СПб. : Питер, 2012. — 784 с.
11. Общие требования к тестовым документам : ГОСТ 2.105-95. – Введ. 01.01.1996. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 1995. – 84 с.
12. Описание программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.402-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 14 с.
13. Текст программы. Требования к содержанию, оформлению и контролю качества : ГОСТ 19.401-2000. – Введ. 01.09.2001. – Минск : Межгос. совет по стандартизации, метрологии и сертификации, 2000. – 16 с.
14. Полное руководство по языку программирования С# 9.0 и платформе .NET 5 [Электронный ресурс] / METANIT.COM. Сайт о программировании. – metanit.com, 2012-2021. – Режим доступа : http://metanit.com/sharp/tutorial. – Дата доступа : 15.04.2021.

# Приложение А

(обязательное)

Текст программы

Создание базы данных

CREATE DATABASE Parking

USE Parking

go

DROP TABLE Operator;

DROP TABLE Payment;

DROP TABLE Report;

DROP TABLE Place;

DROP TABLE Rate;

CREATE TABLE Operator(

id int identity(1,1),

name nchar(15) not null,

surname nchar(15) not null,

pass nchar(18) not null,

CONSTRAINT Prim\_ID\_Admin PRIMARY KEY (id),

)

GO

CREATE TABLE Place(

id int identity(1,1),

status nchar(9) default('Свободно') not null,

CONSTRAINT Prim\_ID\_Place PRIMARY KEY (id),

)

GO

CREATE TABLE Report(

id int identity(1,1),

car nchar(10) not null,

place int not null,

arrival smalldatetime not null,

CONSTRAINT Prim\_ID\_Report PRIMARY KEY (id),

CONSTRAINT Fore\_Place\_Report FOREIGN KEY (place) REFERENCES Place([id])

)

GO

CREATE TABLE Payment(

id int identity(1,1) not null,

departure smalldatetime not null,

pay smallmoney not null,

car nchar(10) not null,

CONSTRAINT Prim\_ID\_Payment PRIMARY KEY (id),

)

GO

CREATE TABLE Rate(

id int identity(1,1) not null,

ddate date not null,

pay smallmoney not null,

CONSTRAINT Prim\_ID\_Rate PRIMARY KEY (id),

)

GO

--INSERT INTO Report(car, place, arrival) values('1234 as-2', 1, ());

--Go

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

INSERT INTO Place(status) values('Свободно');

GO

INSERT INTO Report(car, place, arrival) values ('1', 8, '20-06-2021')

INSERT INTO Rate(ddate, pay) values('12-05-2021', 20);

GO

INSERT INTO Payment(departure, pay, car) values('30-06-2021', 123, '8564 sa-7');

INSERT INTO Payment(departure, pay, car) values('30-06-2021', 231, '2564 sa-7');

GO

INSERT INTO Payment(departure, pay, car) values('01-07-2021', 123, '8564 se-7');

INSERT INTO Payment(departure, pay, car) values('01-07-2021', 133, '8214 se-7');

GO

INSERT INTO Operator(name, surname, pass) values('Адлер', 'Хвяц', '1234');

GO

INSERT INTO Operator(name, surname, pass) values('f', 'f', 'f');

GO

SET DATEFORMAT DMY;

GO

UPDATE Place SET status = 'Занято' where id = 2;

select \* from Rate where id = max(id)

SELECT \* FROM Operator

SELECT \* FROM Place

SELECT \* FROM Report

SELECT \* FROM Payment

SELECT \* FROM Rate

select sum(pay) from Payment where departure >= '01-07-2021' and departure <= '02-07-2021'

Добавление нового администратора

public partial class Reg : Page

{

public Reg()

{

InitializeComponent();

}

private void Continue(object sender, RoutedEventArgs e)

{

string name = loginBox.Text;

string password1 = firstPassBox.Password.Trim();

string password2 = secondPassBox.Password.Trim();

string[] s = name.Split(' ');

bool isGood = true;

//Проверка поля логина

if (name.Length < 3 && name.Length > 20)

{

isGood = false;

loginBox.ToolTip = "Длина логина должна быть от 3 до 20 букв!";

loginBox.Foreground = Brushes.Red;

}

else if(s.Length != 2)

{

loginBox.ToolTip = "Требуется Имя и Фамилия";

loginBox.Foreground = Brushes.Red;

}

else

{

bool stop = false;

foreach (char x in name)

{

if (Char.IsDigit(x))

{

stop = true;

break;

}

}

if (stop)

{

isGood = false;

loginBox.ToolTip = "Недопустимый ввод символов!";

loginBox.Foreground = Brushes.Red;

}

else

{

loginBox.ToolTip = " ";

loginBox.Foreground = Brushes.Black;

}

}

if (password1.Length < 4 && password1.Length > 16)

{

isGood = false;

firstPassBox.ToolTip = "Длина пароля должна быть от 4 до 16 символов!";

firstPassBox.Foreground = Brushes.Red;

}

else

{

firstPassBox.ToolTip = "";

firstPassBox.Foreground = Brushes.Black;

}

//Подтверждение пароля

if (password1 != password2)

{

isGood = false;

secondPassBox.ToolTip = "Пароли не совпадают";

secondPassBox.Foreground = Brushes.Red;

}

else

{

secondPassBox.ToolTip = "";

secondPassBox.Foreground = Brushes.Black;

}

if (isGood == true)

{

SQLbase.Insert($"insert into Operator(name, surname, pass) values (N'{s[0]}',N'{s[1]}',N'{password1}')");

MessageBox.Show("Оператор создан");

NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));

}

}

Расчёт прибыли за отчётный период

private void Calculate(object sender, RoutedEventArgs e)

{

if(startDate.SelectedDate == null || endDate.SelectedDate == null)

{

editButton.Foreground = Brushes.Red;

editButton.ToolTip = "Выберите обе даты";

return;

}

else

{

editButton.Foreground = Brushes.AliceBlue;

editButton.ToolTip = "";

}

DateTime start = startDate.SelectedDate.Value;

DateTime end = endDate.SelectedDate.Value;

using (DataTable Place = SQLbase.Select($"select sum(pay) from Payment where departure >= '{start.Day}-{start.Month}-{start.Year}' and departure <= '{end.Day}-{end.Month}-{end.Year}'"))

{

string str;

if (Place.Rows[0][0].ToString() == null || Place.Rows[0][0].ToString() == "" || Place.Rows[0][0].ToString() == " ")

{

str = "0";

}

else

{

str = String.Format("{0:C}", Place.Rows[0][0]);

}

answerOutput.Content = $"{str}";

}

}

Изменение текущего тарифа

string price = inputPrice.Text;

foreach (Char x in price)

{

if (!char.IsDigit(x))

{

inputPrice.ToolTip = "Требуется числовое значение!";

inputPrice.Foreground = Brushes.Red;

return;

}

else

{

inputPrice.ToolTip = "";

inputPrice.Foreground = Brushes.Black;

}

}

decimal d = Convert.ToDecimal(price);

SQLbase.Insert($"insert into Rate(ddate, pay) values('{DateTime.Today}', {d.ToString().Replace(",",".")})");

MessageBox.Show("Тариф изменён");

Вход администратора

private void Resume(object sender, RoutedEventArgs e)

{

string login = LoginInput.Text.Trim().ToUpper();

string password = PasswordInput.Password;

string[] Fio = login.Split(' ');

bool isGood = true;

if (login.Length < 1 || Fio.Length < 2)

{

isGood = false;

LoginInput.ToolTip = "Введите логин";

LoginInput.Foreground = Brushes.Red;

}

else

{

LoginInput.ToolTip = " ";

LoginInput.Foreground = Brushes.Black;

}

if (password.Length < 1)

{

isGood = false;

PasswordInput.ToolTip = "Введите пароль";

PasswordInput.Foreground = Brushes.Red;

}

else

{

PasswordInput.ToolTip = " ";

PasswordInput.Foreground = Brushes.Black;

}

if (isGood == true)

{

DataTable table = SQLbase.Select($"select name, surname, pass from Operator where name = '{Fio[0]}' and surname = '{Fio[1]}' and pass = '{password}'");

if (table.Rows.Count > 0)

{

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

AdminPage nextPage = new AdminPage(login);

nav.Navigate(nextPage);

}

else

{

LoginInput.ToolTip = "Ошибка входа";

LoginInput.Foreground = Brushes.Red;

PasswordInput.ToolTip = "Ошибка входа";

PasswordInput.Foreground = Brushes.Red;

}

}

Функции окна администратора

public partial class AdminPage : Page

{

private string login;

public AdminPage(string login)

{

this.login = login;

InitializeComponent();

showInfo();

}

private void showInfo()

{

listReports.ItemsSource = SQLbase.Select($"SELECT \* FROM Report").DefaultView;

DateTime s = DateTime.Today;

DateTime s1 = s.AddDays(1);

using (DataTable Place = SQLbase.Select($"select count(\*) from Place where status = 'Занято'")) {

DataTable PlaceCount = SQLbase.Select($"select count(\*) from Place");

ParkPlace.Content = $"{Place.Rows[0][0]}/{PlaceCount.Rows[0][0]}";

}

}

private void Delete(object sender, RoutedEventArgs e)

{

string[] s = login.Split(' ');

SQLbase.Insert($"delete Operator where name = '{s[0]}' and surname = '{s[1]}'");

MessageBox.Show("Аккаунт удалён");

NavigationService.Navigate(new Uri("/MainPage.xaml", UriKind.Relative));

}

private void Reg(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("/Reg.xaml", UriKind.Relative));

}

private void GoToEdit(object sender, RoutedEventArgs e)

{

NavigationService nav;

nav = NavigationService.GetNavigationService(this);

Price nextPage = new Price(login);

nav.Navigate(nextPage);

}

}

Функции окна въезда

public partial class Entrance : Page

{

public Entrance()

{

InitializeComponent();

CheckFreePlace();

}

private void CheckFreePlace()// Проверка присутствия свободных мест

{

using (DataTable table = SQLbase.Select($"select pay from Rate order by id desc"))

{

if (table.Rows.Count == 0)

{

RateOutput.Text = "Бесплатно";

}

else

{

RateOutput.Text = $"{String.Format("{0:C}", table.Rows[0][0])} в час";

}

}

using (DataTable table = SQLbase.Select($"select \* from Place where status = 'Свободно'"))

{

ComboBoxView.ItemsSource = table.DefaultView;

if (table.Rows.Count == 0)

{

//IEnumerable<Button> collection = grid1.Children.OfType<Button>();

//foreach(Button x in collection)

//{

// if(x.Content.ToString() == "Продолжить")

// {

// x.IsVisible = false;

// }

//}

ButtonContinue.Visibility = Visibility.Hidden;

NumberInput.Foreground = Brushes.Red;

NumberInput.IsReadOnly = true;

NumberInput.Text = "Мест нет!";

return;

}

NumberInput.IsReadOnly = false;

ButtonContinue.Visibility = Visibility.Visible;

NumberInput.Foreground = Brushes.Black;

}

}

private void Resume(object sender, RoutedEventArgs e)

{

string num = NumberInput.Text.ToUpper();

bool b = Regex.IsMatch(num, @"^\d{4}\s[A-Z]{2}-+[1-7]$");

bool stations = Regex.IsMatch(num, @"^\d{4}\s[A-Z]{2}-+[0]$");

if (stations)

{

MessageBox.Show("Гос номера не обслуживаем!");

Application.Current.Shutdown();

return;

}

//Проверка правильности ввода значений

if (!b)

{

NumberInput.Foreground = Brushes.Red;

NumberInput.ToolTip = "Неверный формат";

}

else

{

NumberInput.Foreground = Brushes.Black;

NumberInput.ToolTip = "";

}

//if(ComboBoxView.SelectedIndex == -1)

//{

// ComboBoxView.Foreground = Brushes.Red;

// ComboBoxView.ToolTip = "Выберите место";

//}

//else

//{

// ComboBoxView.Foreground = Brushes.Black;

// ComboBoxView.ToolTip = "";

//}

//Занесение значений в бд

if (b && ComboBoxView.SelectedIndex != -1)

{

DataTable tableReport = SQLbase.Select($"select \* from Report where car = '{num}'");

if(tableReport.Rows.Count >= 1)

{

NumberInput.Foreground = Brushes.Red;

NumberInput.ToolTip = "Машина уже присутствует.";

return;

}

else

{

NumberInput.Foreground = Brushes.Black;

NumberInput.ToolTip = "";

}

int index = ComboBoxView.SelectedIndex + 1;

string s = ComboBoxView.Text;

string time = DateTime.Now.ToString();

SQLbase.Insert($"INSERT INTO Report(car, place, arrival) values('{num}', {s}, '{time}');");

SQLbase.Insert($"UPDATE Place SET status = 'Занято' where id = {s}");