Министерство информатизации и связи Республики Татарстан

государственное автономное профессиональное образовательное учреждение

«Международный центр компетенций –

Казанский техникум информационных технологий и связи»

|  |  |
| --- | --- |
| Дипломный проект выполнен и защищен с оценкой \_\_\_\_\_\_\_\_\_ Председатель ГЭК  \_\_\_\_\_\_\_\_\_\_ А.Ф. Хасьянов  «\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г | ДОПУСТИТЬ К ЗАЩИТЕ Заместитель директора по учебной работе \_\_\_\_\_\_\_\_\_\_\_О.С. Тимофеева |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(ДИПЛОМНЫЙ ПРОЕКТ)

Проектирование и разработка программного обеспечения для автосалона вторичных автомобилей

Исполнитель,

студента группы 420 ИСПП \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Ф. Шигапов

*подпись, дата*

Руководитель,

преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.Р. Калинина

*подпись, дата*

г. Казань, 2022 г

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc106046335)

[1 АНАЛИТИЧЕСКАЯ ЧАСТЬ 5](#_Toc106046336)

[1.1 Анализ предметной области 5](#_Toc106046337)

[1.2 Логическая структура данных 6](#_Toc106046338)

[1.3 Выявление связей между объектами информационной системы 7](#_Toc106046339)

[1.4 Обзор существующих аналогов разрабатываемой информационной системы 10](#_Toc106046340)

[2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 13](#_Toc106046341)

[2.1 Диаграмма UML 13](#_Toc106046342)

[2.2 Диаграмма ERD 13](#_Toc106046343)

[2.3 Диаграмма вариантов использования 15](#_Toc106046344)

[2.4 Диаграмма последовательности 16](#_Toc106046345)

[2.5 Диаграмма IDEF0 17](#_Toc106046346)

[3 РЕАЛИЗАЦИЯ ПРОЕКТА 19](#_Toc106046347)

[3.1 Обоснование выбора языка и среды программирования 19](#_Toc106046348)

[3.2 Создание базы данных и API 21](#_Toc106046349)

[3.3 Разработка приложения 24](#_Toc106046350)

[4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА 30](#_Toc106046351)

[4.1 Виды тестирования 30](#_Toc106046352)

[4.2 Тест-кейсы 31](#_Toc106046353)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 34](#_Toc106046354)

[5.1 Установка приложения 34](#_Toc106046355)

[5.2 Работа с программой 34](#_Toc106046356)

[ЗАКЛЮЧЕНИЕ 45](#_Toc106046357)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 47](#_Toc106046358)

[ПРИЛОЖЕНИЕ А (ТЕХНИЧЕСКОЕ ЗАДАНИЕ) 49](#_Toc106046359)

[ПРИЛОЖЕНИЕ В (КОД ПРОГРАММЫ) 53](#_Toc106046360)

# ВВЕДЕНИЕ

В настоящее время во всем мире идет глобальное внедрение информационных технологий во все сферы жизни. Автоматизация какого-либо процесса подразумевает выполнение функций вычислительными машинами, а не человеком. Не исключением является сфера продажи и покупки автомобилей. В условиях роста цен на автомобили не все люди могут позволить себе приобрести новый автомобиль. Также не все люди готовы переплатить за новый автомобиль, либо являются приверженцами автомобилей прошлых поколений выпуска. Поэтому в настоящее время идет рост спроса на поддержанные автомобили. Возникает проблема поиска хорошего поддержанного автомобиля, который не доставлял бы особых хлопот. Существует множество торговых площадок продажи автомобилей с пробегом.

В поиске достойного варианта автомобиля может уйди большое количество времени, за счет того, что в большинстве случаях продаваемы автомобили находиться в разных местах, а также подстроиться с продавцом оказывается достаточно проблематично.

Данная проблема имеет и другую сторону – продажа автомобиля. Не все люди имеют достаточного времени на разговоры и осмотры авто с потенциальными покупателями. Стоить учесть, что существуют типы автомобилей с низкой ликвидностью, и продажа данного автомобиля может оказаться проблемой и займет много времени.

Возникает проблема, где же можно быстро и без особых усилий купить и продать автомобиль. Исходя их этого идет рост популярности автосалона автомобилей с пробегом. Данный типы автосалонов предлагают:

* продажа транспортных средств;
* оценка стоимости авто;
* помощь и консультация во время подбора и покупки машины.

Для ускорения работы автосалона и быстрого получения сведений о транспортном средстве возникает потребность в приложении с единой базой всех автомобилей на складе автосалона.

Но просматривать сведения об автомобилях с настольного компьютера либо носить с собой ноутбук является не самым удобным решением. Куда более удобным решение является работа с системой с мобильного устройства. Разработка мобильного приложения позволить вносить данные об осмотре и фотографии прямо с мобильного приложения и удаленно от рабочего места.

Сферой применения информационной системы являются автосалоны. Информационная система позволит хранить большое количество сведений в одной базе данных, экономить рабочее время за счет автоматизации процессов оценки стоимости автомобиля.

Цель работы – разработать программный продукт, предназначенный для автоматизации процесса отслеживания оценки, бронирования и продажи автомобиля, а также просмотра полного отчета технического состояния автомобиля.

Для достижения указанной цели необходимо решить следующие задачи:

* провести анализ предметной области;
* выполнить разработку пользовательского интерфейса приложения;
* разработать Web API.

Разрабатываемая информационная система предназначена для пользования сотрудниками автосалона.

# 1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

## 1.1 Анализ предметной области

Автосалон подержанных автомобилей занимается выкупом и продажей автомобилей клиентам, и осуществляет свою деятельность за счет перепродажи автомобиле.

Клиент привозит свой автомобиль в автосалон на оценку стоимости. Осмотрщик проводит осмотр автомобиля заполняет отчёт об осмотри, который в свою очередь содержит полную информацию о характеристиках, повреждениях, и общем состояние на бланке. Следующим этапом аналитик просматривает бланк и оглашает цену выкупа клиенту. В том случае если клиента цена устроила, автомобиль выкупают и выставляют его на продажу. После выставления автомобиля на продажу в автосалоне, менеджер может просмотреть отчёт. Если клиенту автомобиль понравился, менеджер осуществляет продажу, в том случае если клиенту необходимо время на обдумывания или поиска недостающей суммы денег, менеджер может забронировать автомобиль на срок не более суток, заполнив на бланке контактные данные клиента.

Целью выпускной квалификационной работы является разработка информационной системы для автоматизации процесса отслеживания процесса оценки, бронирования и продажи автомобиля, а также просмотра полного отчета технического состояния автомобиля.

Для удобного перехода между основными страницами в информационной системе (Профиль, Список автомобилей), требуется разработать снизу экрана меню с вкладками. Для удобного пользования меню необходимо в обозначении вкладок использовать иконки и подпись.

Для удобной работы с профилем пользователя требуется разработать страницу с основными инструментами: редактирование общей информации, и смена пароля. Так для быстрой смены учетной записи пользователей требуется реализация кнопки выхода.

Формирование отчета об осмотре отчёт должно выполнятся посредством заполнения формы, которая в свою очередь содержит поля для заполнения общей информации об транспортном средстве, так же о состоянии лакокрасочного покрытия и технического состояния автомобиля. Неотъемлемой частью отчёта является фотографии самого автомобиля, для быстрого просмотра визуальной части автомобиля. Редактирование отчета должно быть доступно только до момента оценки автомобиля.

Список автомобилей должен отображаться с краткой информации о дате осмотра, госномер, а также статуса автомобиля. Для просмотра отчёта об осмотре, необходимо разработать страницу с полной информацией об авто и информации о проведения осмотра и оценки.

Требуется разработать функцию бронирования автомобиля менеджером, посредством соответствующей кнопки на форме. Список забронированных автомобилей должен выводится в соответствующей вкладке с отображением даты окончания брони. Отмена бронирования должна осуществляться с помощью кнопки на форме информации об автомобиле.

При успешной авторизации требуется реализовать функцию сохранения учетной записи, для автоматической авторизации при повторном запуске приложения.

## 1.2 Логическая структура данных

Логическая структура данных – это представления данных в виде схемы, которая определяет структуру таблиц используемых при организации данных в вычислительных системах и взаимоотношения между ними.

Разрабатываемой информационной системе выделяются следующие сущности:

* Тип кузова (Body: Id, Name);
* Бронирование (Booking: Id, UserId, Date, CarId, FirstName, Name, Phone);
* Марка (Brand: Id, Name);
* Автомобиль (Car: Id, ModelId, YearOfRelease, AnaliticId, InspectorId, EquipmentId, StatusId, VIN, GosNumber, BodyId, Mileage, EngineCapacity, Power, ColorId, CountOwner, TypeTransmissionId, BuyPrice, SellPrice, DateSell, DateBuy, GenerationId, TypeEngineId, FuelTypeId);
* Цвет (Color: Id, Name);
* Тип детали (Detail: Id, Name);
* Привод (Drive: Id, Name);
* Комплектация (Equipment: Id, Name);
* Поколение (Generation: Id, Name);
* Тип топливной систем (FuelType: Id, Name);
* Модель (Model: Id, Name, BrandId);
* Фото автомобиля (PhotoCar: Id, CarId, Picture);
* Информация о повреждения по ЛКП (RepairedPart: Id, DetailId, CarId, Description);
* Роль (Role: Id, Name);
* Статус (Status: Id, Name);
* Тип двигателя (TypeEngine: Id, Name);
* Тип трансмиссии (TypeTransmission: Id, Name);
* Пользователь (User: Id, FirstName, Name, MiddleName, Phone, Password).

Следующим этапом разработки модели данных является указание зависимостей между сущностей.

## 1.3 Выявление связей между объектами информационной системы

Связь указывает отношение между двумя сущностями в виде линии.

Соединение таблиц осуществляется с помощью связывания первичного ключа в родительском и вторичный ключ в дочернем объекте.

Существует три типа связи:

* Много ко многим (М:М) – несколько объектов из одной таблицы соответствует нескольким объектам из другой таблицы;
* Один ко многим (1:М) – одному объекту из одной таблицы соответствует несколько объектов из другой;
* Один к одному (1:1) – один объект из одной таблицы соответствует одному объекту из другой таблицы.

Связи между объектами представлены на таблице 1.1.

Таблица 1.1 - Связи между объектами информационной системы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Объект №1 | Объект №2 | Тип связи | Словесное описание связи |
| 1 | Роль | Пользователь | 1:M | Одинаковую роль могут иметь разные пользователи, но один пользователь может иметь только одну роль |
| 2 | Марка | Модель | 1:М | Один производитель автомобилей имеет несколько моделей авто, но одни модель может иметь только одного производителя |
| 3 | Пользователь | Автомобиль | М:М | Один автомобиль может быть забронированным |
| 4 | Деталь | Информация о повреждения по ЛКП | 1:М | Один тип детали может использоваться в разных записях о ремонте по ЛКП, но в одной записи о ремонте ЛКП может использоваться только одна деталь |
| 5 | Автомобиль | Информация о повреждения по ЛКП | 1:М | Один автомобиль может иметь несколько информаций о ремонтах по ЛКП, но информация о ремонте ЛКП соответствует только одному автомобилю |
| 6 | Автомобиль | Фото автомобиля | 1:М | Один автомобиль может иметь несколько фотографий, но одни фотография может подходить только к одному автомобилю |
| 7 | Кузов | Автомобиль | 1:М | Один тип кузова могут иметь разных автомобили, но один автомобиль имеет только один тип кузова |
| 8 | Цвет | Автомобиль | 1:М | Один цвет может использоваться в разных автомобилях, но один автомобиль моет иметь только один основной цвет |
| 9 | Тип трансмиссии | Автомобиль | 1:М | Один тип трансмиссии может использоваться в разных автомобилях, но в автомобиле может использоваться только один тип трансмиссии |
| 10 | Поколение | Автомобиль | 1:М | Несколько автомобилей могут быть одного поколений, но один автомобиль может иметь только одно поколение |
| 11 | Тип топливной системы | Автомобиль | 1:М | Один тип топливной системы может использоваться в разных автомобилях, но в автомобиле может использоваться только один тип топливной системы |
| 12 | Комплектация | Автомобиль | 1:М | Одну комплектацию могут иметь несколько автомобилях, но один автомобиль может иметь только один тип комплектации |
| 13 | Статус | Автомобиль | 1:М | Один статус могут иметь несколько автомобилях, но один автомобиль иметь только статус |
| 14 | Тип двигателя | Автомобиль | 1:М | Один тип двигателя может использоваться в разных автомобилях, но в автомобиле может использоваться только один тип двигателя |

На основе проведенного анализа предметной области разрабатывается диаграмма базы данных.

## 1.4 Обзор существующих аналогов разрабатываемой информационной системы

На российском рынке существует не малое количество аналогов. Но все они являются комплексными для автомобильной сферы, тем самым повышают затраты на её обслуживание и внедрение.

Одной из самых известных является информационная система 1C:Альфа-Авто. Альфа-Авто – это комплексное решение на базе 1С:Предприятие 8, которое позволяет автоматизировать все бизнес-процессы компаний, занятых в сфере автобизнеса: автосалонов и дилерских центров (Рисунок 1.1).

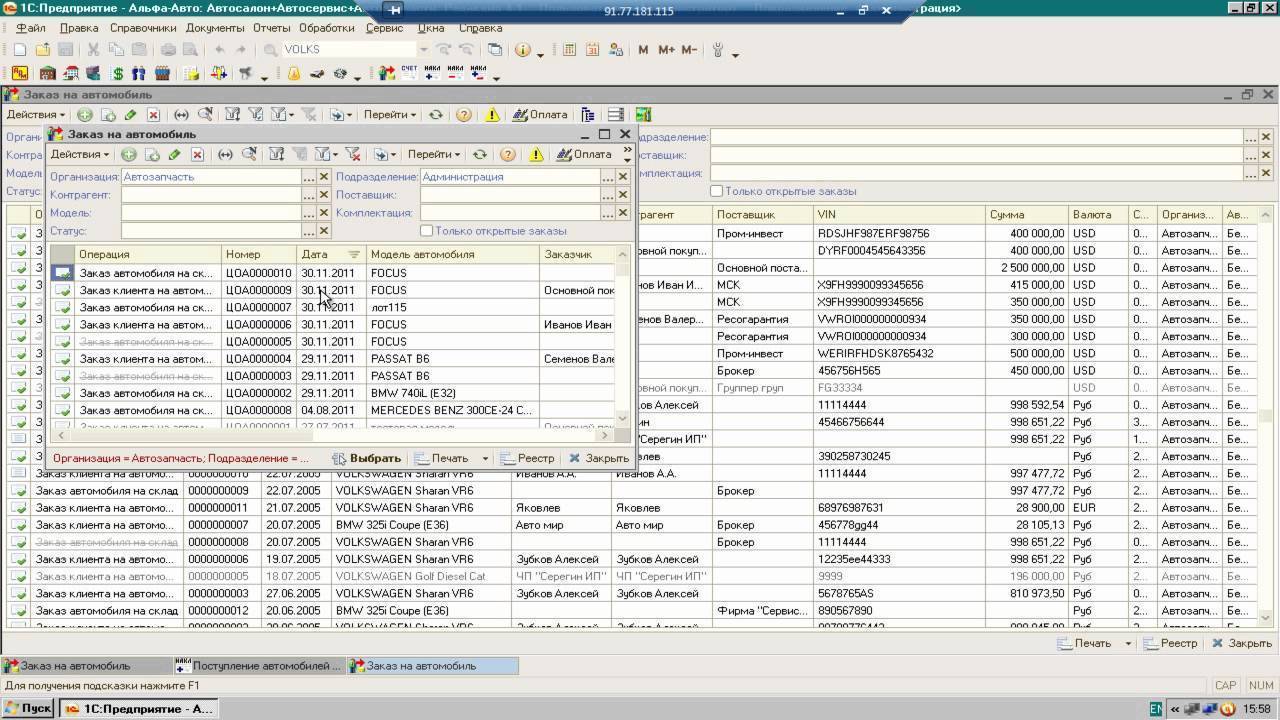


Рисунок 1.1 Интерфейс программы 1С: Альфа-Авто

Преимущества:

* простой и понятный интерфейс;
* регулярные обновления программного продукта;
* обмен данными с бухгалтерскими системами и с каталогами производителей.

Недостатки:

* требуются дополнительный софт для удаленной работы через мобильное приложение;
* избыточный функционал;
* высокая стоимость.

DealerPoint – ещё одна популярна система, помогающая стабилизировать работу автобизнеса и дилерских центров. Система имеет полный функционал для простого и удобного пользования.

Преимущества:

* cинхронизация с телефонией;
* cинхронизация CRM с электронной почтой сотрудников позволит отслеживать историю переписки с потенциальными клиентами прямо из карты контрагента.
* возможность on-line работы сразу несколькими сотрудниками;
* введение статистики по автомобилям и работникам.

Недостатки:

* высокая стоимость;
* отсутствие возможности конфигурирования под определенные задачи.

Исходя из вышеперечисленных аналогов, и указанных недостатков было принято решение реализовать программный продукт, который позволял бы работать удаленно от рабочего места с системой автосалона для ролей оценщика, аналитика, менеджера и администратора.

# 2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 2.1 Диаграмма UML

В ходе реализации проекта может возникнуть необходимость внесения изменений в базу данных или в само приложение. Это может привести к большим временным затратам. Для лучшего понимания концепции приложения, его функционала и уменьшения ошибок в ходе реализации проекта, требуется заблаговременное моделирование информационной системы. Одним из вариантов визуализации приложения является представление его в виде диаграммы. Диаграмма позволяет визуально показать как должна работать программа и его структуру.

Одним из способов моделирования является специализированный язык Unified Modeling Language (UML), имеющий все необходимые графические объекты. Язык UML имеет два типа диаграмм: структурную, отражающая из чего состоит система и поведенческие, в этом типе описывается, что делает система.

Структурной является диаграмма модели данных (ER-модель). ER-модель отображает сущности базы данных и связь между ними. ER моделирование помогает анализировать требования к данным для создания хорошо спроектированной базы данных.

Поведенческими являются диаграммы вариантов использования и последовательности. Диаграммы прецедентов отражают динамическое поведение работающей системы. Он моделирует, как пользователь взаимодействует с системой. Диаграмма последовательности раскрывает структуру системы, позволяет реализовать простой сценарий действий, выполняемых пользователем в системе.

## 2.2 Диаграмма ERD

Диаграмма Entity-Relationship (ERD) предоставляет предварительный просмотр того, как все таблицы должны соединиться, какие поля будут в каждой таблице.

В ER-моделях и моделях данных обычно выделяют до трех уровней детализации: концептуальная, физическая, логическая. Для создания модели подходящим вариантом является концептуальная модель. Концептуальная модель имеет схему с минимальным количеством подробностей в отличии от физической и логической, что позволяет отобразить общую структуру модели (Рисунок 2.1).



Рисунок 2.1 Диаграмма ERD

Для создания модели используют два основных элемента: сущность и связь.  В разрабатываемой диаграмме в роле сущности выступают таблицы: User, Role, Brand, Model, Booking, RepairedParts, Detail, Body, Color, TypeTransmission, Car, Equipment, Status, PhotoCars, FuelType, Generation, TypeEngine.

Данная диаграмма готова для реализации на физическом уровне.

## 2.3 Диаграмма вариантов использования

Диаграмма use case отражает функциональные возможности с использованием действующих лиц и вариантов использования.

Диаграмма use case отображается с помощью таких объектов, как актер, варианта использования, а также стрелка отображающая принадлежность функции к актеру. Актер – это сущность, взаимодействующая с системой. Вариант использования представляет отдельную функциональность системы.

В разрабатываемой информационной системе выделены следующие роли: администратор, осмотрщик, аналитик, менеджер, и следящие функции: добавление, редактирование отчёта об авто, бронирование автомобиля, оценка стоимости автомобиля на основе проведенного осмотра

В данной диаграмме в качестве актера выступает осмотрщик, аналитик, менеджер (Рисунок 2.2).



Рисунок 2.2 Диаграмма Use Case

Таким образом, диаграмма вариантов использования помогает описать систему на концептуальном уровне.

## 2.4 Диаграмма последовательности

В ходе разработки приложения возникает необходимость визуального представления последовательности действий клиента, для выполнения определённой функции программы. Представить их можно с помощью диаграммы последовательности.

Диаграмма последовательности моделирует взаимодействие между объектами в едином сценарии использования. Тем самым иллюстрирует, как взаимодействуют между собой отдельные объекты системы для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования (Рисунок 2.3).

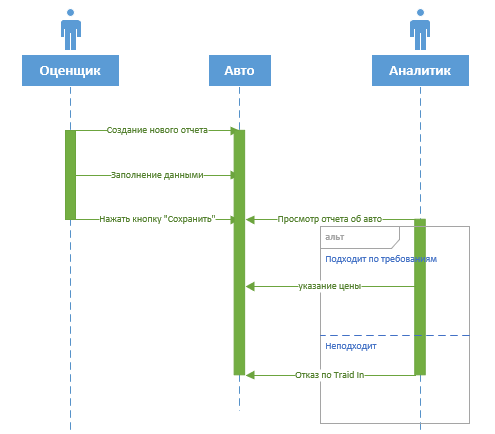


Рисунок 2.3 Диаграмма последовательности

Первым этапом оценщик создает отчёт по средством заполнения информации об автомобиле и нажатия кнопки «Сохранить». Следующим действием аналитик просматривает информации об автомобиле и выносит свой вердикт, если автомобиль подходит по требованиям автосалона и владельца устраивает цена выкупа, то он принимает его и указывает цену выкупа и цену продаж.

Аналогично диаграмма разрабатывается для процесса бронирования и просмотра информации об автомобиле.

## 2.5 Диаграмма IDEF0

IDEF0 – это функциональная модель, которая является ядром построения всех остальных конструкций, она увязывает воедино информационные и материальные потоки, оргструктуру, управляющие воздействия и саму деятельность компании. Графический стандарт для моделирования процессов также принято называть нотацией. То есть нотация – это система требований и правил построения модели деятельности в том или ином виде. Поэтому IDEF0 уместно называть нотацией, входящей в состав методологии SADT.

Центральным элементом модели IDEF0 является функция, которая на схеме отображается в виде функционального блока – прямоугольника, внутри которого указано действие в форме отглагольного существительного. Действие может быть очень разным по масштабу – от деятельности компании вообще и до конкретной манипуляции в частности

Независимо от масштаба действий все функции отображаются единообразно и обязательно содержат 4 ключевых потока, которые жестко закреплены за сторонами функционального блока:

* слева – входы или используемые ресурсы для выполнения функции;
* справа – выходы или результаты выполнения функции;
* сверху – управляющие воздействия, которые определяют, как и сколько нужно произвести результатов;
* снизу – механизмы, которые отражают, кто и с помощью чего должен выполнить эту работу.

Согласно описанию системы одной из функций является создание отчета об осмотре автомобиля. Таким образом, определим работу контекстной диаграммы как «Создание отчета об осмотре», на рисунке 2.4. Далее определим входные и выходные данные, а также механизмы и управление. Для того чтобы произвести создание отчета необходимо заполнить данные автомобиля. В качестве входных данных используется «Характеристики автомобиля», «Фото автомобиля» и «Информация о состоянии автомобиля». Выполнение запроса ведет к добавлению новой записи в базу данных, поэтому выходными данными будет являться «сформированный отчёт». Процесс обработки запросов будет выполняться экраном мобильного устройства.



Рисунок 2.4 Диаграмма IDEF0

Аналогично диаграмма разрабатывается для процессов бронирования, редактирования профиля, оценивания автомобиля и т.д.

# 3 РЕАЛИЗАЦИЯ ПРОЕКТА

## 3.1 Обоснование выбора языка и среды программирования

В настоящее время существует большое количество разновидностей языков для разработки: C, C++, C#, Python, Java.

Язык Си, созданный Денисом Ритчи в начале 70-х годов в Bell Laboratory американской корпорации AT&T, является одним из универсальных языков программирования. Язык Си считается языком системного программирования, хотя он удобен и для написания прикладных программ. Среди преимуществ языка Си следует отметить переносимость программ на компьютеры различной архитектуры и из одной операционной системы в другую, лаконичность записи алгоритмов, логическую стройность программ. Программа, написанная на языке Си, состоит из операторов. Каждый оператор вызывает выполнение некоторых действий на соответствующем шаге выполнения программы.

С++ содержит средства для создания программ практически любого назначения, от низкоуровневых до сложных программных комплексов разного назначения. поддерживает разные технологии и стили программирования, включая объектно-ориентированное, обобщенное. Важным достоинством является предсказуемое выполнение программ, что является важным для систем реального времени.

Java  –  это язык программирования общего назначения, который следует парадигме объектно-ориентированного программирования и подходу «Написать один раз и использовать везде» . Java используется для настольных, сетевых, мобильных и корпоративных приложений. Подробная информация:

Язык С# создан как первый компонентно-ориентированный язык программирования семейства С/С++, является объектно-ориентированным языком, в котором любая сущность представляется объектом, но более простым по сравнению с С++, и сохраняющим его мощь и основные конструкции. Главным преимуществом языка C# перед остальными является, работ на платформе .Net Framework позволяющий удобно работать с базой данных, а также удобство написания кода.

Для написания программного продукта была выбрана среда программирования Visual Studio 2019 и язык разметки приложений XAML.

Visual Studio предоставляет [Конструктор XAML](https://docs.microsoft.com/ru-ru/visualstudio/xaml-tools/creating-a-ui-by-using-xaml-designer-in-visual-studio?view=vs-2019), который помогает создавать пользовательские интерфейсы для приложений Xamarin.Forms. Пользователь может перетаскивать элементы управления из панели инструментов или окна «Ресурсы» и задавать свойства в окне свойств. При выполнении этих действий Visual Studio создает соответствующий код XAML. Если пользователь предпочитает редактировать код XAML напрямую, это также можно сделать.

Для разработки мобильного приложения была выбрана платформа Xamarin.Forms.

Xamarin.Forms – это платформа пользовательского интерфейса с открытым кодом. С помощью Xamarin.Forms разработчики могут создавать приложения для Xamarin.Android, Xamarin.iOS и Windows на основе общей базы кода. Использование язык C# позволяет строить объектно-ориентированную модель и изолировать уровни. Это значит, что можно выстраивать архитектуру с упором на переиспользование кода. Xamarin устраняет все проблемы совместимости оборудования, используя плагины и различные API для работы с функциями общих устройств на всех платформах. Наряду с доступом к особым API-платформам, Xamarin можно компоновать с библиотеками, нативными для конкретной платформы. Это позволяет проводить более тонкую настройку.

База данных обеспечивает хранение информации и представляет собой совокупность данных, организованных по определенным правилам. База данных позволяет структурировать, хранить и обрабатывать данные различного типа.

Система управления базами данных (СУБД) – это совокупность языковых и программных средств, предназначенных для создания, ведения и использования базы данных.

В рамках разрабатываемого проекта используется реляционная база данных, основным преимуществом которой является то, данные хранятся в формате таблиц и строго структурированы. К современным реляционным СУБД относятся Oracle, PostgreSQL, MySQL, Microsoft SQL Server.

PostgreSQL отличается от остальных СУБД тем, что является объектно-ориентированным, это означает что данные моделируются в виде объектов, их атрибутов. Разрабатываемая модель не предусматривает выполнения сложных операций, а данная модель больше предназначена для обработки нескольких операций, и в простых операциях не отличается быстротой обработки.

MySql простая в использовании СУБД. Данная СУБД имеет хорошую производительность как при сложных операциях, так и при простых. Для работы с MySQL в Visual Studio существуют как официальные, так и пользовательские плагины, упрощающие работу с ним.

Специфика работы сервера базы данных MS SQL Server заключается в транзакционной обработке данных. Это означает, что по каждому запросу от СУБД обрабатывается и сохраняется небольшое количество информации, тем самым SQL Server является более подходящим для вставки и удаления данных.

MySQL и SQL Server являются очень похожими по своему синтаксису и параметрам. Ключевым отличием в разработке является возможность работы с средой .NET. Обе СУБД имеют возможность подключения к Visual Studio через Entity Framework, но MySQL в отличие SQL Server для подключения требует установки дополнительных плагинов. В свою же очередь .Net Framework в первую очередь нацелен под работу с SQL Server, тем самым упрощает процесс разработки.

В результате сравнения СУБД SQL Server является самой подходящей при разработке базы данных для данной информационной системы.

## 3.2 Создание базы данных и API

Создание баз данных – процесс создания схемы базы данных и определения необходимых ограничений целостности (соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам).

Основные задачи:

* Обеспечение хранения в БД всей необходимой информации.
* Обеспечение возможности получения данных по всем необходимым запросам.
* Сокращение избыточности и дублирования данных.
* Обеспечение целостности базы данных.

Таблицы создаваемы в БД были выделены в главах выше.

При создании таблицы нужно, чтобы она имела уникальный столбец, которая уникальна для каждой ее строки. Именно по данному столбцу в дальнейшем можно определить какую-нибудь запись.

Такое значение называется первичным ключом таблицы. После того как первичные ключи определены, нужно создать диаграмму. С помощью нее станет возможно определить внешние ключи. Внешний ключ представляет один или несколько столбцов из одной таблицы, который одновременно является потенциальным ключом из другой таблицы.

Для того чтобы определить вторичные ключи, реализованы связи от первичного ключа к предполагаемому вторичному ключу (Рисунок 3.1). После реализации связей автоматически средой разработки присвоились вторичные ключи.

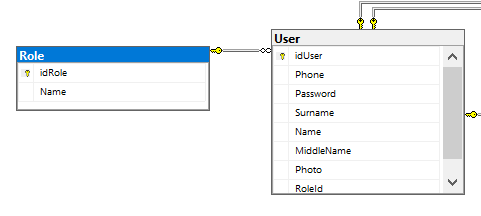


Рисунок 3.1 Определение связей

После определений вторичных ключей база данных готова к работе.

Далее необходимо заполнить каждую таблицу данными, которые будут соответствовать названию столбца и типу данных этого столбца (Рисунок 3.2)

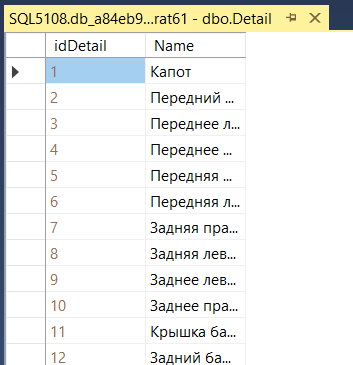


Рисунок 3.2 Заполнение данных

После заполнения данными все таблицы, можно приступить к написанию кода.

Работа с базой данных в разрабатываемом проекте осуществляется через API. Для этого требуется создать пустой проект Web API NET. Framework. В созданном проекте для работы с БД, указывается строка подключения к БД, и автоматически создается модель данных Entities Framework (Рисунок 3.3).

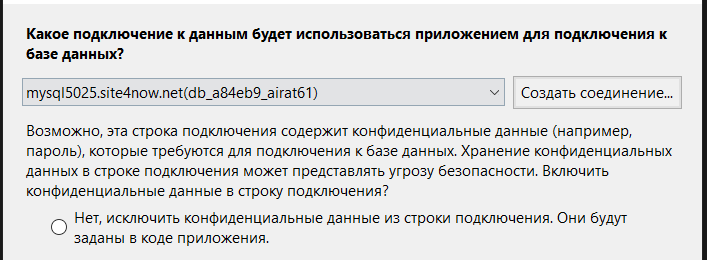


Рисунок 3.3 Подключения к базе данных

Запросы на API обрабатываются с помощью контроллеров. Создаются с помощью правой кнопки мыши по папке Controllers (Рисунок 3.4)

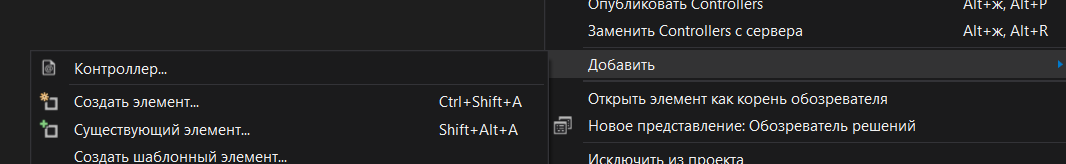


Рисунок 3.4 Открытие окна создание контроллера

В открывшемся окне указывается модель данных, и таблица, для которой создается контроллер (Рисунок 3.5)

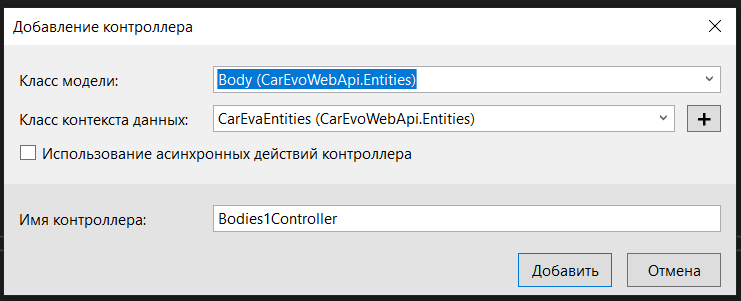


Рисунок 3.5 Создание контроллера

В автоматически созданном классе присутствуют все необходимые методы, осуществляющие удаление добавление обновление данных. Для удаленного доступа к API осуществляется его публикация, по средством ввода данных FTP сервера (Рисунок 3.6).

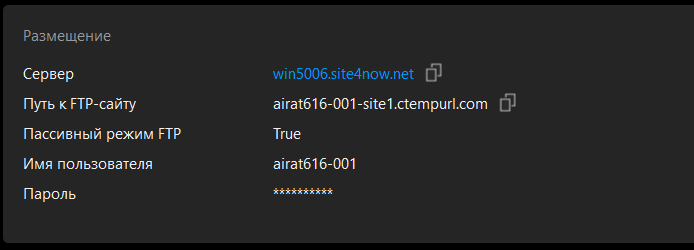


Рисунок 3.6 Размещение API на FTP сервере

После создания базы данных следует этапом разработки приложения.

## 3.3 Разработка приложения

Подготовительным этапом выступает добавление в проект пользовательских шрифтов (Рисунок 3.7).

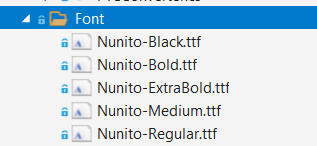


Рисунок 3.7 Пользовательские шрифты

Для использования данных шрифтов требуется их экспортировать в файле AssemblyInfo.cs

Разработка приложения начинается с разработки интерфейса, интерфейс разрабатывается на языке разметки XAML для разделения кода интерфейса и программного кода (Рисунок 3.8).

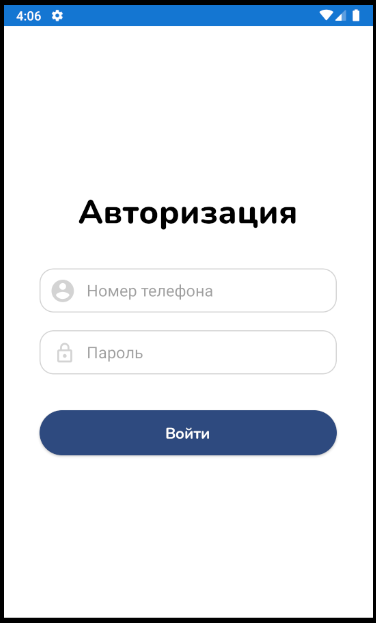


Рисунок 3.8 Страница авторизации

При разработке данного проекта используются такие элементы интерфейса как Picker, Entry, CheckBox, Label, Frame, Button.

Для облегчения создания интерфейса следует использовать стили хранящие в себе общие черты объектов (Рисунок 3.9). Стили позволяют централизованно изменить объекты имущие общий стиль, тем самым позволяет быстро вносить изменения на все элементы, упростить разработку и сократить код.

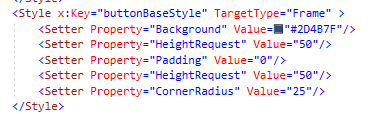


Рисунок 3.9 Style в файле App.xaml

Для удобного перехода по старицам разрабатывается меню вкладок с по (Рисунок 3.10).

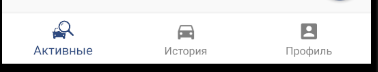


Рисунок 3.10 Меню вкладок

Для разработки меню вкладок использовался элемент TappedPage, TappedPage хранит в себе экземпляры страниц, использующихся в настоящее время, и осуществляет удобный переход по ним с помощью меню вкладок, располагающихся внизу экрана. Добавление страниц осуществляется в конструкторе TappedPage в зависимости от роли (Рисунок 3.11).



Рисунок 3.11 Конструктор страницы InspectorPage

В случае если у клиента отсутствует фотография профиля должна устанавливаться картинка по умолчанию. Для установки картинки по умолчанию используется конвертер. Создание пользовательского конвертера начинается с реализации интерфейса IValueConverter. Данный интерфейс реализует два методы Convert и ConvertBack. В методе Convert указывается условия проверяющие имеется ли у профиля фотографии, в случае отсутствия возвращается ссылка на картинку по умолчанию (Рисунок 3.12).

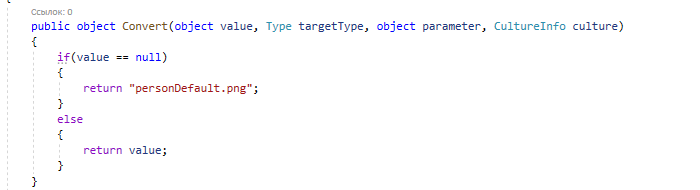


Рисунок 3.12 Конвертер для фотографии клиента

Для удобной работы с данными создаются классы, в которые будет преобразовывать JSON – запрос (Рисунок 3.13).

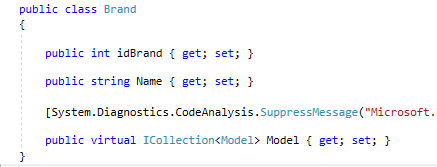


Рисунок 3.13 Класс Brand

Работы осуществляется с помощью класса WebClient. WebClient класс предоставляет общие методы для отправки или получения данных из любого локального, интрасети или интернет-ресурса, определяемого универсальным кодом ресурса (Рисунок 3.14).



Рисунок 3.14 GET-запрос списка пользователей

Ответ от сервера приходит в виде Json и тем самым требует преобразования. Преобразование осуществляется с помощью класса JsonConvert. Для работы с данным классом требуется установить плагин Newtonsoft.Json в диспетчере пакетов NuGet (Рисунок 3.15).

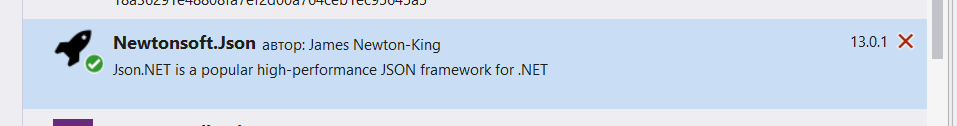


Рисунок 3.15 NuGet пакет Newtonsoft.Json

Навигация осуществляется с помощью класса NavigationService. Для перехода на новую страницу используется метод PushAsync(), принимающий в виде параметра экземпляр страницы, на которую осуществляется переход (Рисунок ). Переход на предыдущую страницу осуществляется с помощью метода PopAsync(Рисунок 3.16).

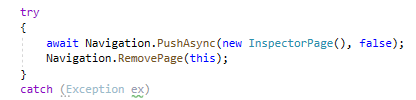


Рисунок 3.16 Функция для перехода cо страницы Authorization на страницу c меню вкладок

Для добавления нескольких фотографий в приложения при составлении отчёта осмотрщиком используется класс CrossMedia позволяющий добавлять несколько фотографий сразу. Для использования данного класса устанавливается плагин Xam.Plugin.Media в диспетчере пакетов NuGet (Рисунок 3.17).

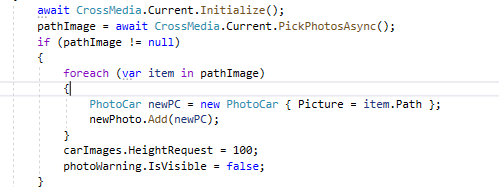


Рисунок 3.17 Функция для добавления картинки

Выбор картинок осуществляется с помощью метод PickPhotosAsync. Данные возвращаются в виде экземпляра класса MediaFile.

Менеджеру доступен номер телефона клиента, забронировавшего автомобиль. В целях повышения удобства работы с приложением необходимо реализовать метод, открывающий приложение для звонка с номером клиента при нажатии на кнопку «Позвонить».

Решить данную задачу позволяет класс CrossMessaging. Для использования данного класса устанавливается плагин Xam.Plugin.Media в диспетчере пакетов NuGet. Открытие приложения осуществляется с помошью метода PhonDialer (Рисунок 3.18).

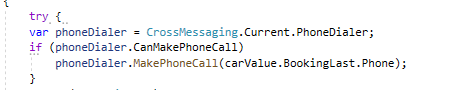


Рисунок 3.18 Функция открытия приложения для звонков

Разрабатываемое приложение предназначается для работы с удаленным сервером, что в свою очередь требует выход в интернет. В целях избежание ошибок в работе приложения при отсутствии сети, необходимо предусмотреть действия при возникновении данных ошибок. Для обработки исключений предназначена конструкция try catch (Рисунок 3.19).

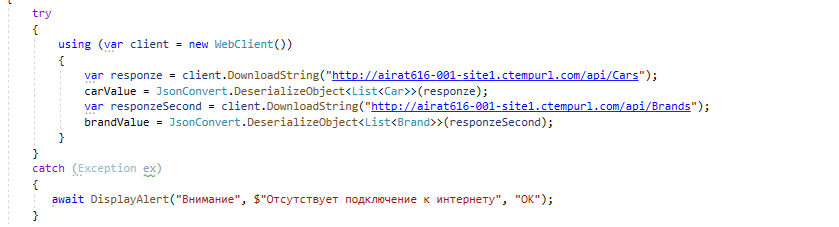


Рисунок 3.19 Обработка исключений

Уведомление пользователя об ошибках осуществляется с помощью метода DisplayAlert принимающий заголовок, содержание сообщения, и название кнопки.

Разработка приложения закончено и готово для последующего тестирования.

# 4 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

## 4.1 Виды тестирования

Тестирование программного обеспечения (ПО) – это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом.

Целью тестирования программного обеспечения является выявление ошибок, пробелов или недостающих требований в отличие от фактических требований. Для тестирования проводится с помощью тест-кейса.

Модульное тестирование выполняется на ранних этапах, когда готовятся отдельные куски приложения (классы, компоненты, функции). В этот момент тестировщики скрупулезно пишут автоматические тесты для каждой функции будущей программы. Это необходимо потому, что проверить «софт» в графическом интерфейсе пока нереально, да и автоматика дает лучший результат.

Интеграционное тестирование проводится на следующем этапе, когда некоторые модули объединяются и превращаются в более крупный компонент, более приближенный к готовой программе. Тестировщики проверяют, как ведут себе ранее разобщенные модули, совмещенные в единый продукт, и как этот готовый продукт функционирует сам по себе. Также на стадии интеграции проверяется совместимость создаваемого ПО с операционной системой, на которой оно будет работать, а иногда еще и с аппаратной частью, чтобы создаваемый продукт нормально функционировал на большем количестве устройств.

Системное тестирования тесно привязана к функциональному и нефункциональному типу. Именно в ходе системного тестирования специально обученные люди проверяют, соответствует ли детище программистов тому, что было заявлено с точки зрения возможностей, и стандартам качества компании с позиции производительности, отзывчивости, отказоустойчивости и прочих технических аспектов.

## 4.2 Тест-кейсы

Тест-кейсы – конкретные действия или наборы действий, выполняемые тестировщиками, чтобы оценить работоспособность ПО. Здесь важно учесть те сценарии, которые будут наиболее близки к реальности.

Для проверки разрабатываемого приложения идеально подходит тест - кейсы, так как не требуют специальных знаний, а так проверяю весь функционал со стороны пользователя.

Любой тест-кейс обязательно включает в себя:

* Уникальный идентификатор тест-кейса – необходим для удобной организации хранения и навигации по нашим тест-наборам.
* Название – основная тема, или идея тест-кейса. Кратное описание его сути.
* Предусловия – описание условий, которые не имеют прямого отношения к проверяемому функционалу, но должны быть выполнены.
* Шаги – описание последовательности действий, которая должна привести нас к ожидаемому результату
* Ожидаемый результат — результат: что мы ожидаем увидеть после выполнения шагов.

Тест-кейсы разрабатывается для основных функций программ:

1. Добавление информации об автомобиле (Таблица 4.1)
2. Авторизация пользователя (Таблица 4.2)
3. Бронирование автомобиля (Таблица 4.3)

Таблица 4.1 – Test-case «Добавление информации об автомобиле»

|  |  |
| --- | --- |
| Уникальный идентификатор и краткое имя | А1 Добавление информации об автомобиле |
| Описание задачи | Осмотрщик вводит данные об автомобиле. |
| Предварительные условия | PRE-1. VIN, Госномер, Год выпуска, Пробег, Объем двигателя, Мощность двигателя, Кол-во хозяев, Компрессия двигателя заполнены в соответствии с валидацией, и Марка, Модель, Поколение, Тип трансмиссии, Тип двигателя, Цвет, Тип кузова, Комплектация, Тип топливной системы, Тип привод, Тип ПТС выбраны их списка, загружено 4 фотографии. |
| Продолжение Таблицы 4.1 | |
| Выходные условия | POST-1. Информация сохраняется.  POST-2. Осуществляется переход на страницу со списком автомобилей.  POST-3. Добавленные данные отображаются на странице «Активные осмотры». |
| Список действий | * Осмотрщик заполняет данные об автомобиле. * Осуществляется переход на страницу со списком автомобилей.   Новый отчет отображается на странице «Активные осмотры». |
| Исключения | Ex-1. Поле VIN не заполнено.   * Информация не сохраняется, и высвечивается сообщение о необходимости заполнения соответствующего поля.   Ex-2. Поля Год выпуска заполнено некорректно.   * Высвечивается информация на экране о необходимости соблюдения валидации.   Ex-3. Отсутствие подключения к сети интернет на мобильном устройстве  Сообщение о отсутствие подключения к сети интернет |

Таблица 4.2 – Test-case «Авторизация с выключенным интернетом»

|  |  |
| --- | --- |
| Уникальный идентификатор и краткое имя | А2 Авторизация пользователя |
| Описание задачи | Пользователь авторизируется в приложении |
| Предварительные условия | PRE-1. Логин и пароль заполнены в соответствие из строк БД. |
| Выходные условия | POST-1. Проверка введённых данных  POST-2. Осуществляется переход на страницу соответствующая роли пользователя |
| Список действий | Заполнить данные авторизации.  Нажать на кнопку «Войти». |
| Исключения | Ex-1. Введение неверного пароля.  Сообщение о неверности введённого пароля  Ex-2. Отсутствие подключения к сети интернет на мобильном устройстве  Сообщение о отсутствие подключения к сети интернет |

Таблица 4.3 – Test-case «Бронирование автомобиля»

|  |  |
| --- | --- |
| Уникальный идентификатор и краткое имя | А3 Бронирование автомобиля |
| Описание задачи | Менеджер бронирует автомобиль за определенным клиентом |
| Предварительные условия | PRE-1. Фамилия и Имя клиента заполнено, номер телефона заполнено в соответствие с валидацией. |
| Выходные условия | POST-1. Проверка на корректность введённых данных  POST-2. Осуществляется переход на страницу с информацией об автомобиле с отображение информации о брони. |
| Список действий | * Выбрать автомобиль. * Нажать на кнопку «Забронировать». * Заполнить данные в соответствие с валидацией.   Нажать кнопку «Сохранить». |
| Исключения | Ex-1. Введение некорректного номера телефона.   * Сообщение о несоответствие формата номер телефона   Ex-2. Отсутствие подключения к сети интернет на мобильном устройстве   * Сообщение о отсутствие подключения к сети интернет   Ex-3. Отсутствие подключения к сети интернет на мобильном устройстве  Сообщение о отсутствие подключения к сети интернет |

Тест-кейсы – основная почва для проведения тестирования. Зачастую проекты отказываются от тест-кейсов, остающихся на лидирующих позициях, из-за существенных недостатков и самым главным «Время=деньги», отдавая предпочтение расширенным чек листам.

В результате тестирования выявленные ошибки были исправлены.

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Установка приложения

На мобильное устройство необходимо скачать файл «carEva.apk», находящийся в папке с проектом (Рисунок 5.1). Скаченный проект требуется запустить, и тем самым запустится установщик приложения. В открывшемся окне требуется разрешить действие и дождаться полной установки (Рисунок 5.1). Установленное приложения появится на рабочем столе или в списке установленных приложения.

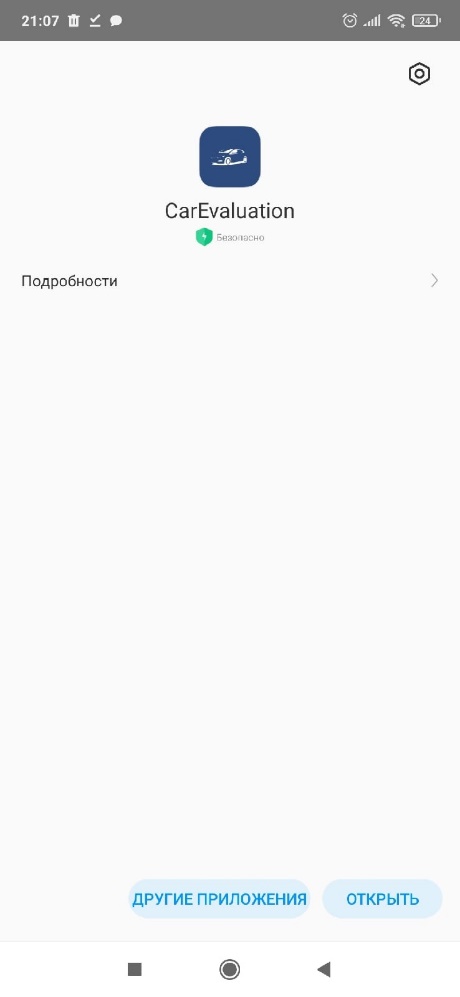


Рисунок 5.1 Установка приложения

## Работа с программой

После запуска приложения открывается страница авторизации (Рисунок 5.1). Для в входа в систему необходимо авторизироваться путем ввода номера телефона, пароля и нажатия на кнопку «Войти». При нажатии на кнопку осуществляется проверка на корректность ввода данных.

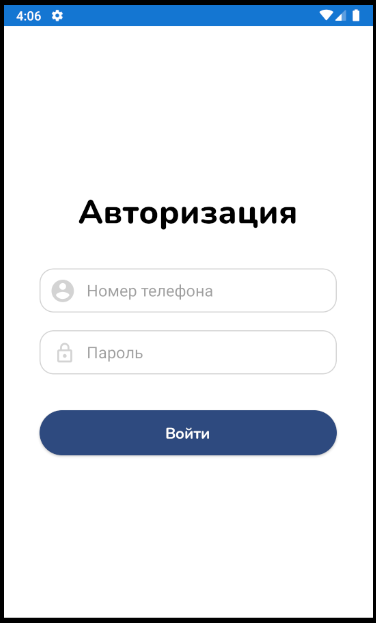


Рисунок 5.2 Страница авторизации

Если отсутствует подключение к интернету высвечивается соответствующее сообщение с выбором либо закрыть приложение, либо повторить попытку авторизации (Рисунок 5.3).

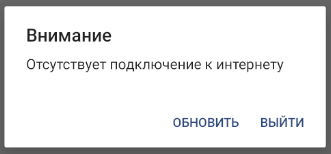


Рисунок 5.3 Сообщение об отсутствие соединения

В случае если данные пользователь зарегистрирован в система осуществляется переходи на страницу соответствующая его роли, если же пользователь не найден, то высвечивается соответствующая подсказка.

После авторизации открывается старицы с меню вкладок в нижней части экрана (Рисунок 5.4).

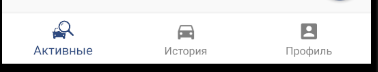


Рисунок 5.4 Навигационное меню

У всех ролей данные вкладки отличаются, но общей является вкладка «Профиль» (Рисунок 5.5). На данной странице отображается фамилия, имя, фотография и должность пользователя. Для редактирования профиля необходимо нажать на вкладку «Редактировать профиль».

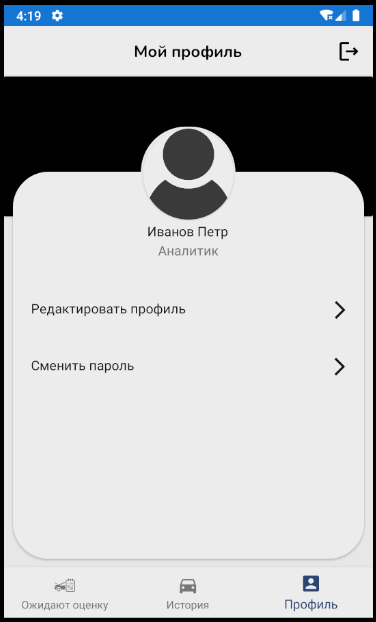


Рисунок 5.5 Страница «Мой профиль»

В открывшейся странице отображается ФИО и номер телефона пользователя (Рисунок 5.6).

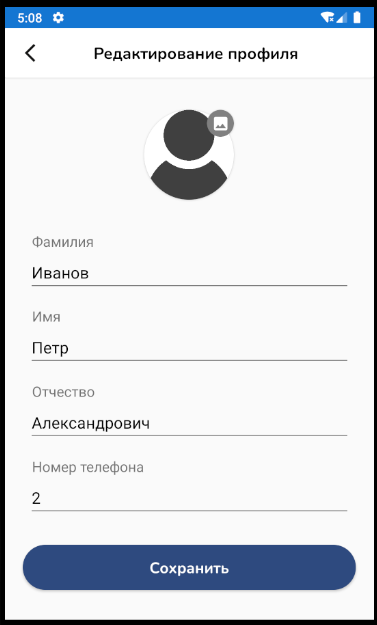


Рисунок 5.6 Страница редактирования профиля

Добавление фотографии пользователя осуществляется с помощью нажатия на кнопку добавления в правом верхнем углу иконки пользователя. В открывшемся проводнике необходимо выбрать нужную фотография.

После редактирования для сохранения изменений необходимо нажать кнопку сохранить, если необходимо вернуться на страницу профиля без сохранения изменений необходимо нажать на иконку в виде стрелки в верхним левом углу (Рисунок 5.7).

****

Рисунок 5.7 Кнопка для перехода нп предыдущую страницу

Редактирования пароля осуществляется переходом на страницу с помощью нажаться на вкладку «Редактирование пароля». В отрывшемся окне необходимо ввести старый пароль, новый пароль и нажать на кнопку «Сохранить» (Рисунок 5.8).

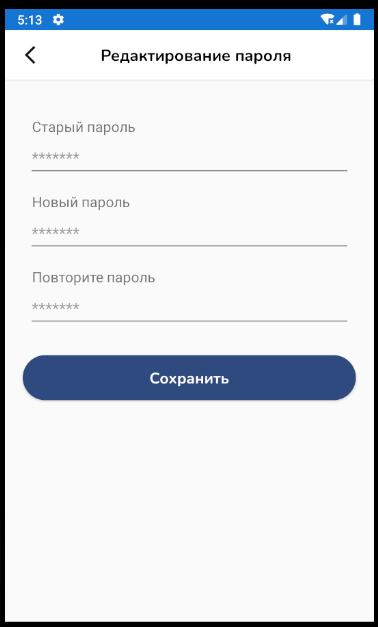


Рисунок 5.8 Страница редактирования пароля

В случае если старый пароль не верный высвечивается соответствующее сообщение. При успешной смене пароля осуществляется переход назад.

Выход из учетной записи осуществляется нажатием на иконку (Рисунок 5.9) в правом верхнем углу страницы.



Рисунок 5.9 Кнопка для выхода их учетной записи

Стоит отметить, что для удобного и быстрого поиска автомобилей существует поиск по госномеру в начале каждого списка. Поиск срабатывает автоматически при вводе каждого символа (Рисунок 5.10).

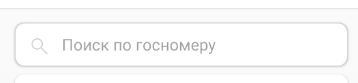


Рисунок 5.10 Поле «Поиск»

В приложении существует 4 роли, рассмотрим возможности каждой по отдельности. При авторизации за роль осмотрщика открывается страница со списком активных осмотров, здесь высвечивается список отчётов об автомобиле которые находятся в статусе «Ожидают оценку» (Рисунок 5.11).

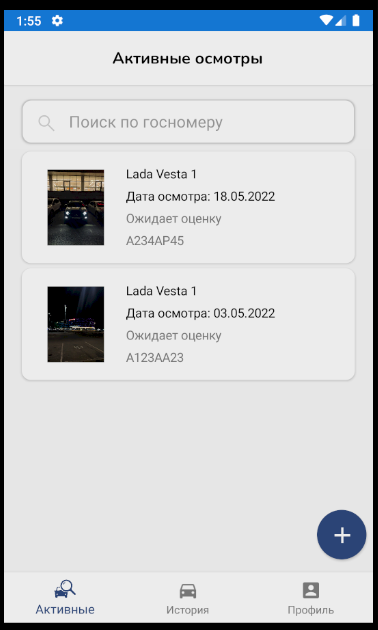


Рисунок 5.11 Список автомобилей, ожидающих оценку

На следующей вкладки «История» высвечивается список автомобилей, которые получили заключение осмотрщика или уже проданы. Для просмотра информации об автомобиле необходимо нажать на сам элемент в списке и осуществится переход на страницу с информацией (Рисунок 5.12).

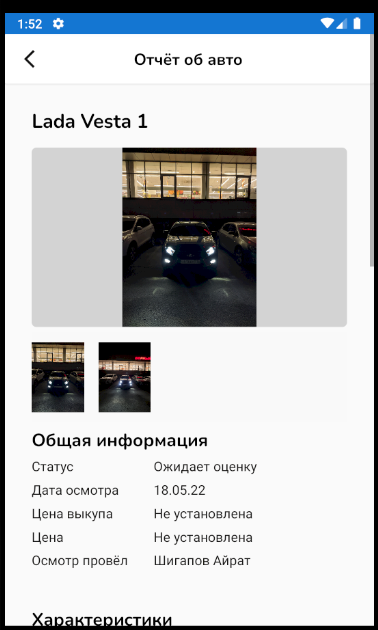


Рисунок 5.12 Отчёт об автомобиле

Редактирование отчёта об автомобиле осуществляется на странице редактирования, переход на который осуществляется с помощью кнопки редактировать на странице об информации об автомобиле. Добавление и редактирование осуществляется на одной форме. Для добавления необходимо нажать на кнопку (Рисунок 5.13) на странице списка активных осмотров.



Рисунок 5.13 Кнопка добавления нового отчёта

После редактирования/добавления информации об автомобиле для сохранения необходимо нажать на кнопку «Сохранить». При некорректности введённых данных высветится советующее сообщение (Рисунок 5.14).

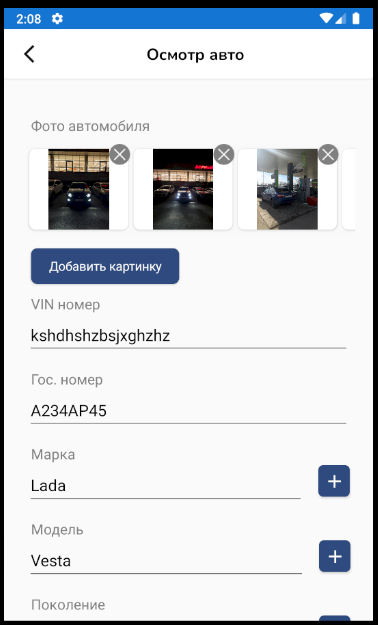


Рисунок 5.14 Кнопка добавления нового отчёта

В случает если требуется отменить редактирование или добавления, нажатием на кнопку назад высвечивается сообщение с необходимостью подтверждения действий (Рисунок 5.15).

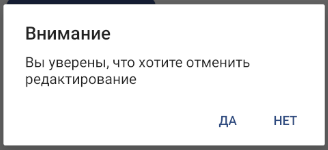


Рисунок 5.15 Сообщение для подтверждения действий

При авторизации за роль оценщика открывается страница со списком автомобилей ожидающие оценку. На вкладке «История оценок» отображается список автомобилей прошедшие оценку. Для выставления стоимости автомобиля необходимо перейти на страницу с информации об авто нажатием на сам элемент в списке. На странице «Информация об авто» в отличие от пользователей других ролей появляются кнопки «Отклонить» и «Принять» (Рисунок 5.16).

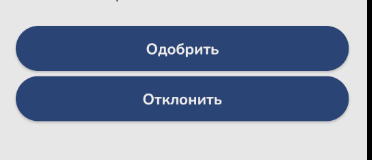


Рисунок 5.16 Кнопки управления отчётам у роли оценщика

После нажатия кнопки отклонить, автомобиль отклоняется для выкупа по Traid-IN и осуществляется переход назад, на страницу списка автомобилей, ожидающих оценку. Нажатием на кнопку «Принять» открывается страница для указания цены выкупа и цены продажи (Рисунок 5.17).

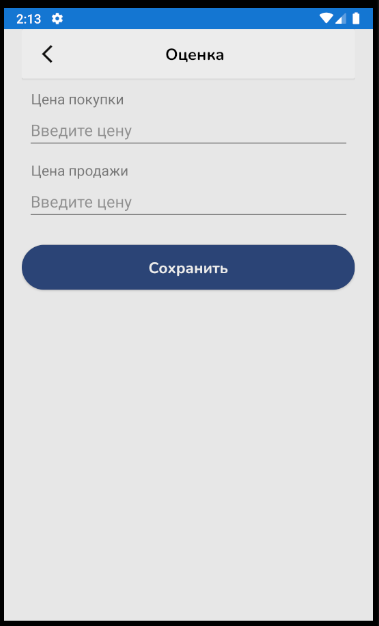


Рисунок 5.17 Кнопки управления отчётам о

Роль менеджера предусматривает просмотр информации об автомобиле, бронирования и продажи. У менеджера доступны две вкладки автомобилей, находящихся в продажи, а также забронированные за ним автомобили. Стоит отметить, что с списке забронированных авто в краткой информации вместо статуса выступает срок окончания бронирования (Рисунок 5.18).

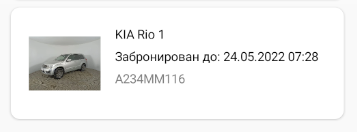


Рисунок 5.18 Краткая информация о забронированном авто

На странице просмотра информации об автомобиле в конце появляются кнопки «Забронировать» и «Продать» (Рисунок 5.19).

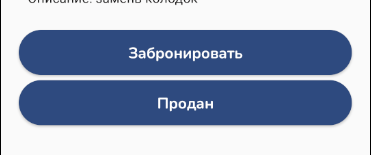


Рисунок 5.19 Кнопки управления отчётом у роли менеджера

Кнопка продать меняет статус автомобиля на продан и автомобиль пропадает со списка автомобилей в наличие. Нажатием на кнопку «Забронировать» отрывается форма для заполнения информации о клиенте, который бронирует автомобиль (Рисунок 5.20).

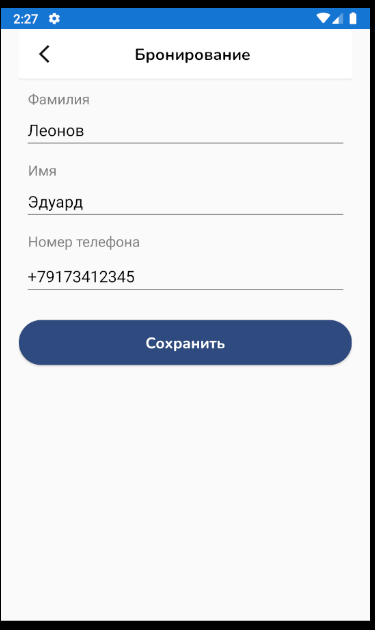


Рисунок 5.20 Форма бронирования авто

После сохранения в странице с информацией об авто добавления информация о бронирования доступная всем пользователем, а также кнопка «Позвонить» и «Отменить бронь» (Рисунок 5.21).

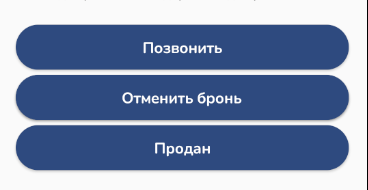


Рисунок 5.21 Кнопки управления забронированным авто у роли менеджера

Нажатием «Отменить бронь» текущая бронь отменяется. Стоит отметить, что бронирование автомобиль осуществляется на срок – 24 часа. Кнопка «Позвонить» открывает приложение по умолчанию телефона для звонков с указанным номером клиента (Рисунок 5.22).

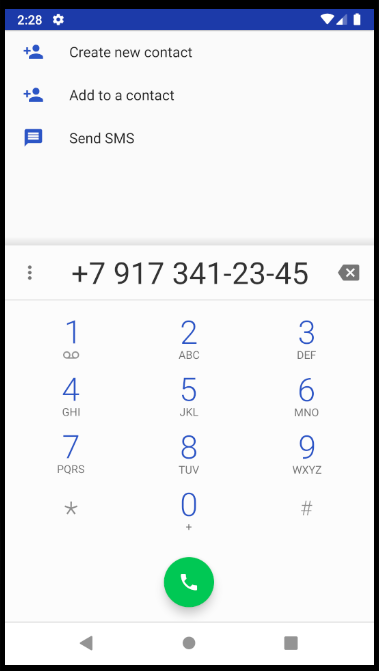


Рисунок 5.22 Форма звонка клиенту

Администратор может просматривать информацию об автомобиля в зависимости от стадии на которой они находятся (Рисунок 5.23).

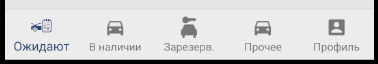


Рисунок 5.23 Меню вкладок Администратора

На вкладке прочее находятся автомобиле которые либо уже проданы, либо было отказано по Traid-IN.

# ЗАКЛЮЧЕНИЕ

Результатом дипломного проекта является создание информационной системы для автосалона поддержанных автомобилей, основным назначением которой является оценка и выкуп автомобилей по Traid-IN, а также последующей продажи. Разработанный информационная система помогает систематизировать учёт автомобилей в автосалоне.

Информационная система разработана с использованием наиболее актуальных элементов управления. Информационная система получил красивый и понятный интерфейс. Посредством чего взаимодействие пользователя с программным продуктом оказалось простым. Информационная система разрабатывалась для мобильного устройства, и тем самым позволяет работать сотрудникам автосалона удаленно от рабочего места

В ходе разработки информационной системы для автосалона поддержанных автомобилей были приобретены практические навыки анализа предметной области, логической и физической разработки, проанализированы функциональные возможности автосалонов, отдельные компоненты применены на практике.

При выполнении проекта были пройдены все этапы разработки специализированного прикладного программного обеспечения:

* постановка задачи;
* сбор необходимых исходных данных;
* разработка и отладка программного продукта.

С помощью СУБД MS SQL Server была разработана структура базы данных, предоставляющая всю информацию в удобном для просмотра виде, позволяя быстро ориентироваться в дальнейшей работе с ней.

Для работы с базой данных MS SQL Server была разработана удалённая API, позволяющая оправлять, обновлять, получать и удалять данные в базе данных, а также хранить картинки.

Для разработки приложения была выбрана платформа Xamarin Forms позволяющая удобно работать Web API, был спроектирован консольный интерфейс, который обеспечивает удобную работу клиента магазина с приложением. Интерфейс реализован в виде страниц, в которых есть наборы элементов, с помощью которых можно взаимодействовать с базой данных автосалона.

В результате было получено мобильное приложение имеющий весь необходимы функционал для процесса осмотра, оценки и продажи автомобиля.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Структуры данных и основные алгоритмы их обработки./ Т. Н. Варфоломеева.: ФЛИНТА, 2018. – 256 c.

2. Введение в разработку приложений для ОС Android./ Березовская Ю.В., Юфрякова О.А., Вологдина В.Г.: НОУ "ИНТУИТ", 2019. – 434 с.

3. **Паттерны объектно-ориентированного проектирования**/ **Эрих Гамма, Ричард Хелм, Ральф Джонсон, Джон Влиссидес: Питер**, 2021. – 448 с.

4. **Чистый код: создание, анализ и рефакторинг**/Роберт Мартин: Питер, 2021. – 464 с.

5. Введение в тестирование программного обеспечения / Пол Амманн, Джефф Оффатт: Cambridge University Press, 2019. – 473;

6. Microsoft SQL Server 2019: A Beginner's Guide / Дусан Петкович: McGraw-Hill Education, 2020. – 864с.

7. **Рефакторинг. Улучшение проекта, существующего кода**/ **Мартин Фаулер**: Вильямс, 2019. – 448 с.

8. Разработка обслуживаемых программ на языке C# / Джус Виссер: пер. с англ. Р. Н. Рагимова: ДМК Пресс, 2017. – 192 с.

9. Проектирование веб-API / Пер. с англ. Д. А. Беликова: ДМК Пресс, 2020.– 440 с.

10. ГОСТ 2.105-95 межгосударственный стандарт. Единая система конструкторской документации общие. Требования к текстовым документам

11. Правила оформления технического задания. [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/507414/. – (Дата обращения: 10.05.2022)

12. Руководства по Xamarin.Forms. [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/xamarin/get-started/quickstarts/.– (Дата обращения: 03.05.2022)

13. Placeholder для элемента Picker. [Электронный ресурс]. – Режим доступа: https://www.c-sharpcorner.com/article/placeholder-for-picker-xamarin-forms/.– (Дата обращения: 04.05.2022)

14. Разработка пользовательского элемента Entry. [Электронный ресурс]. – Режим доступа: https://www.c-sharpcorner.com/article/custom-entry-using-xamarin-forms/.– (Дата обращения: 04.05.2022)

15. Простое руководство по UML-диаграммам и моделированию баз данных. [Электронный ресурс]. – Режим доступа: https://www.microsoft.com/ru-ru/guide-to-uml-diagramming-and-database-modeling.– (Дата обращения: 03.05.2022)

16. Руководства по веб - Api. [Электронный ресурс]. – Режим доступа: https://docs.microsoft.com/ru-ru/aspnet/web-api/overview/. – (Дата обращения: 03.05.2022)

17. Руководство по создания диаграммы IDEF0. [Электронный ресурс]. – Режим доступа: https://support.microsoft.com/ru-ru/office/создание-схемы-idef0-ea7a9289-96e0-4df8-bb26-a62ea86417fc.– (Дата обращения: 05.05.2022)

18. Руководства по написанию Test Case. [Электронный ресурс]. – Режим доступа: https://testengineer.ru/kak-pisat-test-kejsy-polnoe-rukovodstvo/.– (Дата обращения: 09.05.2022).

19. Проектирование диаграммы последовательности UML (sequence diagram) [Электронный ресурс]. – Режим доступа: https://nationalteam.worldskills.ru/skills/proektirovanie-diagrammy posledovatelnosti-uml-sequence-diagram/ (дата обращения: 03.05.2022)

20. Оформление списка литературы по ГОСТу [Электронный ресурс]. – Режим доступа: https://исторический-сайт.рф/Оформление-списка-литературы-по-ГОСТу-1.html (дата обращения: 01.06.2022).

# ПРИЛОЖЕНИЕ А (ТЕХНИЧЕСКОЕ ЗАДАНИЕ)

1. НАЗВАНИЕ ПРОГРАММЫ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Наименование работы: Автоматизированная информационная система для автосалона вторичных автомобилей. Информационная система предназначена для хранения информации об автомобилях выкупленных по Traid – IN.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Основание для разработки является задание выданное учебным учреждением «МЦК-КТИТС»

Плановые сроки начала работы: 07.04.22

Плановый срок окончания работы: 19.06.22

3. НАЗНАЧЕНИЕ РАЗРАБОТКИ

Данная информационная система предназначена для решения круга задач, которые возникают при выкупе автосалоном автомобиля по Traid-IN. Данная информационная система предназначена для работников автосалона.

Функциональные требования:

Программный продукт должен иметь обеспечивать выполнение следующих функций:

* просмотр информации об автомобиле;
* добавление новых автомобилей и изменять существующие;
* установку цены для автомобиля;
* бронирование автомобиля
* добавления картинок
* возможность открытия приложения для звонков с приложения автосалона
* валидацию полей;
* редактирование профиля;
* возможность записи только на свободные временные интервалы.

4. ТРЕБОВАНИЕ К ПРОГРАММЕ И ПРОГРАММНОМУ ИЗДЕЛИЮ

4.1 Требования к функциональным характеристикам.

Разрабатываемая модель должна обладать следующими функциями:

* Работать под управлением Android.
* Иметь доступный и простой интерфейс пользователя.
* Иметь гибкую систему настроек.
* Серверная часть должна хранить базу данных пользователей, имеющих доступ к системе и обеспечивать аутентификацию пользователей согласно имеющихся записей.

4.2 Требования к надежности.

Система должна работать без перебоев, даже при большом количестве пользователей, а также защищать от ошибочных действий пользователей на уровне пользовательского приложения;

4.3 Условия эксплуатации.

Стандартные условия эксплуатации программных продуктов. Необходимые сотрудники для обслуживания серверной части системы – системный администратор для обслуживания собственно сервера.

4.4 Требования к составу и параметрам технических средств.

Мобильное устройство имеющая подключение к сети.

Компьютера для СУБД, включающий в себя:

* процессор x86 с тактовой частотой, не менее 1 ГГц;
* оперативную память объемом, не менее 1 Гб;
* постоянную память не менее 500 Гб

4.5 Требования к информационной и программной совместимости.

Требуется выход в интернет для соединения с БД.

4.6 Требования к маркировке и упаковке.

Не предъявляются.

4.7 Требования к транспортированию и хранению.

Не предъявляются.

4.8 Специальные требования.

Не предъявляются.

5. ТРЕБОВАНИЕ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Разрабатываемая система должна включать справочную информацию о работе системы. В состав сопровождающей документации должны входить: описание системы, руководство пользователя.

6. СТАДИИ И ЭТАПЫ РАЗРАБОТКИ

Разработка должна быть проведена в 4 стадии:

1. Разработка технического задания

На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:

1) постановка задачи;

2) определение и уточнение требований к техническим средствам;

3) определение требований к информационной системе;

4) определение стадий, этапов и сроков разработки информационной системы и документации на неё;

2. Разработка проектной документации

На этапе разработки проектной документации должны быть выполнены перечисленные ниже работы:

1) определение основных вариантов использования;

2) проектирование структуры базы данных в виде (ER диаграммы);

3) проектирование диаграммы последовательности

4) проектирование структуры пользовательского интерфейса;

5) согласование и утверждение проектной документации.

3. Рабочее проектирование

На этапе разработки должна быть выполнена работа по разработке информационной системы на основе проектной документации, кодированию и отладке. На стадии рабочего проектирования должны быть выполнены перечисленные ниже этапы работ:

1) разработка информационной системы;

2) разработка СУБД.

3) тестирование

4. Ввод в действие

На этапе подготовки и передачи программы должна быть выполнена работа по подготовке и передаче программы и программной документации в эксплуатацию.

7. ПОРЯДОК КОНТРОЛЯ И ПРИЕМКИ СИСТЕМЫ

Испытание системы производится путем проверки всех функциональных требований. Производится проверка соответствия экранных форм описанию в руководстве для пользователя.

На основании протокола испытаний исполнитель совместно с заказчиком подписывают акт приемки-сдачи программы в эксплуатацию.

# ПРИЛОЖЕНИЕ В (КоД ПРОГРАММЫ)

AddNewItemPage.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="CarEvaluation.AddNewItemPage">

<ContentPage.Content>

<Grid x:Name="stack" >

<Grid.RowDefinitions>

<RowDefinition Height="55"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Frame Padding="0" HeightRequest="55" Grid.Row="0">

<StackLayout VerticalOptions="Center"

HorizontalOptions="FillAndExpand"

Orientation="Horizontal">

<Frame Padding="0" HasShadow="False">

<Path Style="{StaticResource nextIconStyle}"

HorizontalOptions="Start"

Margin="20,0,0,0"

WidthRequest="20"

Rotation="180"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoBack\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Label x:Name="AppBar"

Text="Осмотр авто"

HorizontalOptions="CenterAndExpand"

FontSize="18"

Margin="0,0,30,0"

Padding="0"

Style="{StaticResource mainTextStyle}"/>

</StackLayout>

</Frame>

<StackLayout Orientation="Vertical" Grid.Row="1" Margin="20,0,20,0">

<StackLayout x:Name="brandEdit" IsVisible="False" Spacing="2">

<Label Text="Марка"

Margin="5,5,0,0"

Style="{DynamicResource infoFieldStyle}"/>

<Picker x:Name="brandPicker"

TextColor="Black"

HorizontalOptions="FillAndExpand"

ItemDisplayBinding="{Binding Name}"

Title="Выберите марку авто"/>

<Label x:Name="brandWarning" Text="Марка не выбрана" Style="{StaticResource warningStyle}"/>

</StackLayout>

<Entry x:Name="fieldEntry" TextChanged="fieldEntry\_TextChanged" Margin="0,10,0,0" Style="{StaticResource entryBaseStyle}">

</Entry>

<Label x:Name="labelWarning" Text="Заполните поле" Style="{StaticResource warningStyle}"/>

<Frame x:Name="Save"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

VerticalOptions="Start"

Margin="0,20,0,100">

<Label Text="Сохранить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoSave\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</Grid>

</ContentPage.Content>

</ContentPage>

AddNewItemPage.xaml.cs:

public partial class AddNewItemPage : ContentPage

{

private int valueType;

VehicleReportPage vhp = new VehicleReportPage(null);

List<Model> models;

public AddNewItemPage(int z, VehicleReportPage vhp)

{

NavigationPage.SetHasNavigationBar(this, false);

InitializeComponent();

valueType = z;

this.vhp = vhp;

OnLoad();

}

private async void OnLoad()

{

try

{

if (valueType == 1)

{

AppBar.Text = "Добавление новой марки";

fieldEntry.Placeholder = "Введите название марки";

}

else if (valueType == 2)

{

AppBar.Text = "Добавление новой модели";

fieldEntry.Placeholder = "Введите название модели";

brandEdit.IsVisible = true;

using (var client = new WebClient())

{

string responze = "";

responze = client.DownloadString(string.Format("{0}/Models", MainInfoClass.apiDBSource));

models = JsonConvert.DeserializeObject<List<Model>>(responze);

}

brandPicker.ItemsSource = vhp.brandCollection.ToList();

}

else if (valueType == 3)

{

AppBar.Text = "Добавление нового цвета";

fieldEntry.Placeholder = "Введите название цвета";

}

else if (valueType == 4)

{

AppBar.Text = "Добавление новой комплектации";

fieldEntry.Placeholder = "Введите название комплектации";

}

else if (valueType == 5)

{

AppBar.Text = "Добавление нового поколения";

fieldEntry.Placeholder = "Введите название поколения";

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private async void GoBack\_Tapped(object sender, EventArgs e)

{

await Navigation.PopAsync(false);

}

private async void GoSave\_Tapped(object sender, EventArgs e)

{

try

{

using (var client = new WebClient())

{

if (valueType == 1)

{

string apiBD = string.Format("{0}/Brands", MainInfoClass.apiDBSource);

var responze = client.DownloadString(apiBD);

List<Brand> brands = JsonConvert.DeserializeObject<List<Brand>>(responze);

if (brands.ToList().Where(x => x.Name == fieldEntry.Text).FirstOrDefault() != null)

{

labelWarning.Text = "Данная марка уже существует";

labelWarning.IsVisible = true;

return;

}

else

{

client.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string html = client.UploadString(apiBD, JsonConvert.SerializeObject(new Brand { Name = fieldEntry.Text }));

var newBrand = JsonConvert.DeserializeObject<Brand>(html);

vhp.brandCollection.Add(newBrand);

}

}

else if (valueType == 2)

{

if (brandPicker.SelectedItem == null)

{

brandWarning.IsVisible = true;

return;

}

if (models.Where(x => x.Name == fieldEntry.Text).FirstOrDefault() != null)

{

labelWarning.Text = "Данная модель уже существует";

labelWarning.IsVisible = true;

return;

}

else

{

Brand brandValue = brandPicker.SelectedItem as Brand;

Model newModel = new Model { Name = fieldEntry.Text, BrandId = brandValue.idBrand };

client.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string html = client.UploadString(string.Format("{0}/Models", MainInfoClass.apiDBSource), JsonConvert.SerializeObject(newModel));

Model eq = JsonConvert.DeserializeObject<Model>(html);

if (brandValue.Model == null)

{

brandValue.Model = new List<Model>();

}

brandValue.Model.Add(eq);

if (vhp.modelCollection.Count != 0 && (vhp.modelCollection.FirstOrDefault() as Model).BrandId == (brandPicker.SelectedItem as Brand).idBrand)

{

vhp.modelCollection.Add(eq);

}

}

}

else if (valueType == 3)

{

string apiBD = string.Format("{0}/Colors", MainInfoClass.apiDBSource);

var responze = client.DownloadString(apiBD);

List<ColorCar> colors = JsonConvert.DeserializeObject<List<ColorCar>>(responze);

if (colors.Where(x => x.Name == fieldEntry.Text.Trim()).FirstOrDefault() != null)

{

labelWarning.Text = "Данный цвет уже существует";

labelWarning.IsVisible = true;

return;

}

else

{

ColorCar сColor = new ColorCar { Name = fieldEntry.Text.ToString() };

client.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string html = client.UploadString(apiBD, JsonConvert.SerializeObject(сColor));

vhp.colorCollection.Add(JsonConvert.DeserializeObject<ColorCar>(html));

}

}

else if (valueType == 4)

{

string apiBD = string.Format("{0}/Equipments", MainInfoClass.apiDBSource);

var responze = client.DownloadString(apiBD);

List<Equipment> equipments = JsonConvert.DeserializeObject<List<Equipment>>(responze);

if (equipments.Where(x => x.Name == fieldEntry.Text).FirstOrDefault() != null)

{

labelWarning.Text = "Данная комплектация уже существует";

labelWarning.IsVisible = true;

return;

}

else

{

client.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string html = client.UploadString(apiBD, JsonConvert.SerializeObject(new Equipment { Name = fieldEntry.Text }));

vhp.equipmentCollection.Add(JsonConvert.DeserializeObject<Equipment>(html));

}

}

else if (valueType == 5)

{

string apiBD = string.Format("{0}/Generations", MainInfoClass.apiDBSource);

var responze = client.DownloadString(apiBD);

List<Generation> generations = JsonConvert.DeserializeObject<List<Generation>>(responze);

if (generations.ToList().Where(x => x.Name.Trim() == fieldEntry.Text.Trim()).FirstOrDefault() != null)

{

labelWarning.Text = "Данное поколение уже существует";

labelWarning.IsVisible = true;

return;

}

else

{

client.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string html = client.UploadString(apiBD, JsonConvert.SerializeObject(new Generation { Name = fieldEntry.Text }));

vhp.generationCollection.Add(JsonConvert.DeserializeObject<Generation>(html));

}

}

}

await Navigation.PopAsync(false);

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

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

{

labelWarning.IsVisible = false;

}

}

Authorization.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:CarEvaluation"

x:Class="CarEvaluation.Authorization">

<Grid Background="White">

<StackLayout Padding="40,0,40,0" VerticalOptions="CenterAndExpand" Spacing="20">

<Label Text="Авторизация" Style="{DynamicResource labelBaseStyle}"/>

<Frame x:Name="frameLogin"

Padding="2"

Margin="0,20,0,0"

HasShadow="False"

BorderColor="LightGray"

CornerRadius="15">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Path x:Name="loginIcon" Fill="LightGray" VerticalOptions="Center" HorizontalOptions="Center" Margin="8,1,0,0"

Data="M12.5 0C5.6 0 0 5.6 0 12.5C0 19.4 5.6 25 12.5 25C19.4 25 25 19.4 25 12.5C25 5.6 19.4 0 12.5 0ZM12.5 3.75C14.575 3.75 16.25 5.425 16.25 7.5C16.25 9.575 14.575 11.25 12.5 11.25C10.425 11.25 8.75 9.575 8.75 7.5C8.75 5.425 10.425 3.75 12.5 3.75ZM12.5 21.5C9.375 21.5 6.6125 19.9 5 17.475C5.0375 14.9875 10 13.625 12.5 13.625C14.9875 13.625 19.9625 14.9875 20 17.475C18.3875 19.9 15.625 21.5 12.5 21.5Z"/>

<local:RoundedEntry x:Name="loginEntry"

Grid.Column="1"

Placeholder="Номер телефона"

Margin="-15,0,0,0"

Keyboard="Telephone"

Focused="login\_Focused"

Unfocused="login\_Unfocused"

HorizontalOptions="FillAndExpand"/>

</Grid>

</Frame>

<Label x:Name="loginWarning" Text="Пользователь не найден" Style="{StaticResource warningStyle}"/>

<Frame x:Name="framePassword"

Padding="2"

HasShadow="False"

BorderColor="LightGray"

CornerRadius="15">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Path x:Name="passwordIcon" Fill="LightGray" VerticalOptions="Center" HorizontalOptions="Center" Margin="8,0.5,0,0"

Data="M18.7503 8.33334H17.7087V6.25C17.7087 3.375 15.3753 1.04167 12.5003 1.04167C9.62532 1.04167 7.29199 3.375 7.29199 6.25V8.33334H6.25033C5.10449 8.33334 4.16699 9.27084 4.16699 10.4167V20.8333C4.16699 21.9792 5.10449 22.9167 6.25033 22.9167H18.7503C19.8962 22.9167 20.8337 21.9792 20.8337 20.8333V10.4167C20.8337 9.27084 19.8962 8.33334 18.7503 8.33334ZM9.37532 6.25C9.37532 4.52083 10.7712 3.125 12.5003 3.125C14.2295 3.125 15.6253 4.52083 15.6253 6.25V8.33334H9.37532V6.25ZM18.7503 20.8333H6.25033V10.4167H18.7503V20.8333ZM12.5003 17.7083C13.6462 17.7083 14.5837 16.7708 14.5837 15.625C14.5837 14.4792 13.6462 13.5417 12.5003 13.5417C11.3545 13.5417 10.417 14.4792 10.417 15.625C10.417 16.7708 11.3545 17.7083 12.5003 17.7083Z"/>

<local:RoundedEntry x:Name="passwordEntry"

Grid.Column="1"

Focused="passwordEntry\_Focused"

Unfocused="passwordEntry\_Unfocused"

Placeholder="Пароль"

Margin="-15,0,0,0"

IsPassword="True"

HorizontalOptions="FillAndExpand"/>

</Grid>

</Frame>

<Label x:Name="passWarning" Text="Пароль неверный" Style="{StaticResource warningStyle}"/>

<Frame Style="{StaticResource buttonBaseStyle}"

Margin="0,20,0,0"

Padding="0">

<Label Text="Войти" Style="{DynamicResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoAuthorization\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</Grid>

</ContentPage>

Authorization.xaml.cs:

public partial class Authorization : ContentPage

{

List<User> user = new List<User>();

public Authorization()

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

}

protected override void OnAppearing()

{

base.OnAppearing();

ConnectToApi();

}

private void ConnectToApi()

{

try

{

using (var client = new WebClient())

{

var responze = client.DownloadString(string.Format("{0}/Users",MainInfoClass.apiDBSource));

user = JsonConvert.DeserializeObject<List<User>>(responze);

var responzeSecond = client.DownloadString(string.Format("{0}/Status", MainInfoClass.apiDBSource));

MainInfoClass.statusList = JsonConvert.DeserializeObject<List<Status>>(responzeSecond);

}

int z = Preferences.Get("id", 0);

MainInfoClass.userList = user;

MainInfoClass.user = user.Where(x => x.idUser == z).FirstOrDefault();

if (MainInfoClass.user != null)

{

TransitionPage();

}

}

catch (Exception ex)

{

ChechConnect();

}

}

private async void ChechConnect()

{

if (await DisplayAlert("Внимание", $"Отсутствует подключение к интернету", "Выйти", "Обновить") == false)

{

ConnectToApi();

}

else

{

System.Diagnostics.Process.GetCurrentProcess().Kill();

}

}

private void login\_Focused(object sender, FocusEventArgs e)

{

frameLogin.BorderColor = Color.FromHex("2D4B7F");

loginIcon.Fill = new SolidColorBrush(Color.FromHex("2D4B7F"));

}

private void login\_Unfocused(object sender, FocusEventArgs e)

{

if (loginEntry.Text == null || loginEntry.Text.ToString().Length == 0)

{

frameLogin.BorderColor = Color.LightGray;

loginIcon.Fill = Brush.LightGray;

}

}

private void passwordEntry\_Focused(object sender, FocusEventArgs e)

{

framePassword.BorderColor = Color.FromHex("2D4B7F");

passwordIcon.Fill = new SolidColorBrush(Color.FromHex("2D4B7F"));

}

private void passwordEntry\_Unfocused(object sender, FocusEventArgs e)

{

if (passwordEntry.Text == null || passwordEntry.Text.ToString().Length == 0)

{

framePassword.BorderColor = Color.LightGray;

passwordIcon.Fill = Brush.LightGray;

}

}

private async void GoAuthorization\_Tapped(object sender, EventArgs e)

{

if (String.IsNullOrWhiteSpace(loginEntry.Text) == true)

{

loginWarning.Text = "Заполните поле";

loginWarning.IsVisible = true;

}

else if (String.IsNullOrWhiteSpace(passwordEntry.Text) == true)

{

loginWarning.Text = "Заполните поле";

passWarning.IsVisible = true;

}

else

{

MainInfoClass.user = user.Where(x => x.Phone == loginEntry.Text).FirstOrDefault();

if (MainInfoClass.user == null)

{

loginWarning.Text = "Пользователь не найден";

loginWarning.IsVisible = true;

}

else if (MainInfoClass.user.Password != passwordEntry.Text)

{

passWarning.Text = "Пароль неверный";

passWarning.IsVisible = true;

}

else

{

Preferences.Set("id", MainInfoClass.user.idUser);

TransitionPage();

}

}

}

private async void TransitionPage()

{

try

{

await Navigation.PushAsync(new InspectorPage(), false);

Navigation.RemovePage(this);

}

catch (Exception ex)

{

await DisplayAlert("Внимание", $"Отсутствует подключение к интернету", "ОК");

}

}

}

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

CarApprovalPage.xaml:

x:Class="CarEvaluation.Pages.CarApprovalPage">

<ContentPage.Content>

<ScrollView Orientation="Vertical" Grid.Row="0" Grid.RowSpan="2">

<StackLayout Orientation="Vertical" Margin="20,0,20,0" Grid.Column="0" Grid.ColumnSpan="2">

<Frame Grid.Row="0" Padding="0" HeightRequest="55">

<StackLayout VerticalOptions="Center"

HorizontalOptions="FillAndExpand"

Orientation="Horizontal">

<Frame Padding="0" HasShadow="False">

<Path Style="{StaticResource nextIconStyle}"

HorizontalOptions="Start"

Margin="20,0,0,0"

WidthRequest="20"

Rotation="180"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoBack\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Label Text="Оценка"

HorizontalOptions="CenterAndExpand"

FontSize="18"

Margin="0,0,30,0"

Padding="0"

Style="{StaticResource mainTextStyle}"/>

</StackLayout>

</Frame>

<StackLayout Spacing="2">

<Label Text="Цена покупки"

Margin="10,5,0,0"

Style="{DynamicResource infoFieldStyle}"/>

<Entry x:Name="buyEntry"

Style="{StaticResource entryBaseStyle}"

Placeholder="Введите цену"

Keyboard="Numeric"

TextChanged="buyEntry\_TextChanged"

MaxLength="17"

TextColor="Black"

Margin="6,0,6,0"/>

<Label x:Name="buyWarning"

Text="Цена не указана"

Style="{StaticResource warningStyle}"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Цена продажи"

Margin="10,5,0,0"

Style="{DynamicResource infoFieldStyle}"/>

<Entry x:Name="sellEntry"

Style="{StaticResource entryBaseStyle}"

Placeholder="Введите цену"

Keyboard="Numeric"

TextChanged="sellEntry\_TextChanged"

MaxLength="17"

TextColor="Black"

Margin="6,0,6,0"/>

<Label x:Name="sellWarning"

Text="Цена не указана"

Style="{StaticResource warningStyle}"/>

</StackLayout>

<Frame x:Name="Save"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

VerticalOptions="StartAndExpand"

Margin="0,20,0,0">

<Label Text="Сохранить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Save\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

CarApprovalPage.xaml.cs:

public partial class CarApprovalPage : ContentPage

{

Car carValue;

public CarApprovalPage(Car car)

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

this.carValue = car;

if (car.StatusId == 2)

{

buyEntry.Text = car.BuyPrice.ToString();

sellEntry.Text = car.SellPrice.ToString();

}

}

private async void Save\_Tapped(object sender, EventArgs e)

{

try

{

bool war = true;

if (String.IsNullOrWhiteSpace(buyEntry.Text))

{

buyWarning.IsVisible = true;

war = false;

}

if (String.IsNullOrWhiteSpace(sellEntry.Text))

{

sellWarning.IsVisible = true;

war = false;

}

if (war)

{

carValue.BuyPrice = Math.Round(Convert.ToDecimal(buyEntry.Text), 2);

carValue.SellPrice = Math.Round(Convert.ToDecimal(sellEntry.Text));

carValue.AniliticId = MainInfoClass.user.idUser;

carValue.user1 = MainInfoClass.user;

carValue.StatusId = 2;

carValue.Status = MainInfoClass.statusList.Where(x=>x.idStatus == 2).FirstOrDefault();

int indexCar = carValue.idCar;

string z = string.Format("{0}/Cars/{1}", MainInfoClass.apiDBSource, indexCar);

using (WebClient wc = new WebClient())

{

wc.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string json = JsonConvert.SerializeObject(carValue);

wc.UploadString(z, WebRequestMethods.Http.Put, json);

}

(MainInfoClass.carInspector as IRefreshListCar).RefreshData();

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", $"Отсутствует подключение к интернету", "ОК");

}

}

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

{

buyWarning.IsVisible = false;

}

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

{

sellWarning.IsVisible = false;

}

private async void GoBack\_Tapped(object sender, EventArgs e)

{

if (await App.Current.MainPage.DisplayAlert("Внимание", "Вы уверены, что хотите отменить редактирование", "Нет", "Да") == false)

{

await Navigation.PopAsync(false);

}

}

}

CarInspector.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:CarEvaluation"

x:Class="CarEvaluation.CarInspector">

<ContentPage.Content>

<Grid HorizontalOptions="FillAndExpand">

<Grid.RowDefinitions>

<RowDefinition />

<RowDefinition Height="70"/>

</Grid.RowDefinitions>

<ScrollView Orientation="Vertical" Grid.Row="0" Grid.RowSpan="2">

<StackLayout Orientation="Vertical" Grid.Column="0" Grid.ColumnSpan="2">

<Frame Grid.Row="0" Padding="0" HeightRequest="60">

<Label x:Name="titleApp" Text="Установка цены"

HorizontalOptions="Center"

VerticalOptions="Center"

FontSize="18"

Padding="0"

TextColor="Black"

FontFamily="NunitoBold"/>

</Frame>

<Frame x:Name="searchFrame"

Padding="2"

Margin="20,10,20,0"

HasShadow="False"

BorderColor="LightGray"

CornerRadius="10">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="40"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Path x:Name="searchIcon" Fill="LightGray" VerticalOptions="Center" HorizontalOptions="Center" Margin="8,1,0,0"

Data="M19.9 20.975L13.325 14.4C12.825 14.8333 12.2417 15.1708 11.575 15.4125C10.9083 15.6542 10.2 15.775 9.45 15.775C7.65 15.775 6.125 15.15 4.875 13.9C3.625 12.65 3 11.1417 3 9.37501C3 7.60834 3.625 6.10001 4.875 4.85001C6.125 3.60001 7.64167 2.97501 9.425 2.97501C11.1917 2.97501 12.6958 3.60001 13.9375 4.85001C15.1792 6.10001 15.8 7.60834 15.8 9.37501C15.8 10.0917 15.6833 10.7833 15.45 11.45C15.2167 12.1167 14.8667 12.7417 14.4 13.325L21 19.875L19.9 20.975ZM9.425 14.275C10.775 14.275 11.925 13.7958 12.875 12.8375C13.825 11.8792 14.3 10.725 14.3 9.37501C14.3 8.02501 13.825 6.87084 12.875 5.91251C11.925 4.95417 10.775 4.47501 9.425 4.47501C8.05833 4.47501 6.89583 4.95417 5.9375 5.91251C4.97917 6.87084 4.5 8.02501 4.5 9.37501C4.5 10.725 4.97917 11.8792 5.9375 12.8375C6.89583 13.7958 8.05833 14.275 9.425 14.275Z"/>

<local:RoundedEntry x:Name="searchEntry"

Grid.Column="1"

Placeholder="Поиск по госномеру"

Margin="-15,0,0,0"

FontFamily="NunitoRegular"

Focused="search\_Focused"

TextChanged="searchEntry\_TextChanged"

Unfocused="search\_Unfocused"

HorizontalOptions="FillAndExpand"/>

</Grid>

</Frame>

<RefreshView x:Name="refreshView">

<StackLayout>

<CollectionView x:Name="listCar" >

<CollectionView.ItemTemplate>

<DataTemplate>

<Grid HeightRequest="110" Margin="20,0,20,20" >

<Frame HeightRequest="80"

BackgroundColor="White"

CornerRadius="10"

Margin="20,3,20,3"

HorizontalOptions="FillAndExpand">

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoInfoCar\_Tapped"/>

</Frame.GestureRecognizers>

<Grid VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="80"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Image Source="{Binding FirstImage}" HorizontalOptions="FillAndExpand" Grid.Column="0"/>

<StackLayout Orientation="Vertical" VerticalOptions="Center" Margin="10,-5,0,0" Grid.Column="1">

<Label Style="{StaticResource infoCarStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1} {2}">

<Binding Path="Model.Brand.Name"/>

<Binding Path="Model.Name"/>

<Binding Path="Generation.Name"/>

</MultiBinding>

</Label.Text>

</Label>

<Label Text="{Binding DateBuy, StringFormat='{}Дата осмотра: {0:dd.MM.yyyy}'}" Style="{StaticResource infoCarStyle}"/>

<Label Text="{Binding Status.Name}" TextColor="Gray" Style="{StaticResource infoCarStyle}"/>

<Label Text="{Binding GosNumber}" TextColor="Gray" Style="{StaticResource infoCarStyle}"/>

</StackLayout>

</Grid>

</Frame>

</Grid>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

<CollectionView x:Name="listCarReserv" IsVisible="False" >

<CollectionView.ItemTemplate>

<DataTemplate>

<Grid HeightRequest="110" Margin="20,0,20,20" >

<Frame HeightRequest="80"

BackgroundColor="White"

CornerRadius="10"

Margin="20,3,20,3"

HorizontalOptions="FillAndExpand">

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoInfoCar\_Tapped"/>

</Frame.GestureRecognizers>

<Grid VerticalOptions="FillAndExpand" HorizontalOptions="FillAndExpand">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="80"/>

<ColumnDefinition Width="\*"/>

</Grid.ColumnDefinitions>

<Image Source="{Binding FirstImage}" HorizontalOptions="FillAndExpand" Grid.Column="0"/>

<StackLayout Orientation="Vertical" Margin="10,0,0,0" Grid.Column="1">

<Label Style="{StaticResource infoCarStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1} {2}">

<Binding Path="Model.Brand.Name"/>

<Binding Path="Model.Name"/>

<Binding Path="Generation.Name"/>

</MultiBinding>

</Label.Text>

</Label>

<Label Text="{Binding BookingDate, StringFormat='{}Забронирован до: {0:dd.MM.yyyy HH\\:mm}'}" Style="{StaticResource infoCarStyle}"/>

<Label Text="{Binding GosNumber}" TextColor="Gray" Style="{StaticResource infoCarStyle}"/>

</StackLayout>

</Grid>

</Frame>

</Grid>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

</StackLayout>

</RefreshView>

</StackLayout>

</ScrollView>

<Frame x:Name="AddButton"

CornerRadius="50"

HeightRequest="55"

WidthRequest="55"

IsVisible="False"

Margin="0,0,7,7"

VerticalOptions="Start"

BackgroundColor="{StaticResource BluGr}"

HorizontalOptions="End"

Padding="0"

Grid.Row="1">

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="InsertCarInspector\_Tapped"/>

</Frame.GestureRecognizers>

<Path Fill="White"

VerticalOptions="Center"

Margin="0,0,4,4"

HorizontalOptions="Center" Data="M23.75 16.25H16.25V23.75H13.75V16.25H6.25V13.75H13.75V6.25H16.25V13.75H23.75V16.25Z"/>

</Frame>

</Grid>

</ContentPage.Content>

</ContentPage>

CarInspector.xaml.cs:

public partial class CarInspector : ContentPage, IRefreshListCar

{

internal List<Car> carValue = new List<Car>();

internal List<Car> carAll = new List<Car>();

internal ObservableRangeCollection<Car> carObc;

List<Brand> brandValue;

int loadValue = 0;

string apiBDCars = string.Format("{0}/Cars", MainInfoClass.apiDBSource);

string apiBDBrands = string.Format("{0}/Brands", MainInfoClass.apiDBSource);

int ndx = 6;

public CarInspector()

{

InitializeComponent();

LoadCar();

carValue = carValue.OrderByDescending(x => x.DateBuy).ToList();

refreshView.Command = RefreshCommand;

foreach (var item in carValue)

{

item.Model.Brand = brandValue.Where(x => x.idBrand == item.Model.BrandId).FirstOrDefault();

}

}

private async void LoadCar()

{

try

{

using (var client = new WebClient())

{

var responze = client.DownloadString(apiBDCars);

carValue = JsonConvert.DeserializeObject<List<Car>>(responze);

var responzeSecond = client.DownloadString(apiBDBrands);

brandValue = JsonConvert.DeserializeObject<List<Brand>>(responzeSecond);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", $"Отсутствует подключение к интернету", "ОК");

}

}

public CarInspector(bool z) : this()

{

GetCarAnalit();

loadValue = 1;

titleApp.Text = "Ожидают оценку";

}

public CarInspector(bool z, int r) : this()

{

GetCarActivInspect();

loadValue = 2;

titleApp.Text = "Активные осмотры";

AddButton.IsVisible = true;

}

public CarInspector(int r, bool z) : this()

{

GetCarInspect();

loadValue = 6;

titleApp.Text = "История осмотров";

}

public CarInspector(int z) : this()

{

GetCar();

loadValue = 3;

titleApp.Text = "Автомобили";

}

public CarInspector(string z) : this()

{

GetCarManager();

loadValue = 4;

listCarReserv.IsVisible = true;

listCar.IsVisible = false;

titleApp.Text = "Забронированные авто";

}

public CarInspector(string z, bool r) : this()

{

GetCarHistory();

loadValue = 5;

titleApp.Text = "История оценок";

}

public CarInspector(string z, int r) : this()

{

GetCarOther();

loadValue = 7;

titleApp.Text = "Другие";

}

protected override void OnAppearing()

{

base.OnAppearing();

}

void GetCarOther()

{

carAll = carValue.Where(x => x.StatusId == 3 || x.StatusId == 4).ToList();

listCar.ItemsSource = carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

}

void GetCarHistory()

{

carAll = carValue.Where(x => x.AniliticId == MainInfoClass.user.idUser).ToList();

listCar.ItemsSource = carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

}

void GetCar()

{

carAll = carValue.Where(x => x.StatusId == 2 && x.BookingDate < DateTime.Now).ToList();

listCar.ItemsSource = carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

}

void GetCarAnalit()

{

carAll = carValue.Where(x => x.StatusId == 1).ToList();

listCar.ItemsSource = carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

}

void GetCarActivInspect()

{

carAll = carValue.Where(x => x.InspectorId == MainInfoClass.user.idUser && x.StatusId == 1).ToList();

listCar.ItemsSource = carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

}

void GetCarInspect()

{

carAll = carValue.Where(x => x.InspectorId == MainInfoClass.user.idUser && x.StatusId != 1).ToList();

listCar.ItemsSource = carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

}

void GetCarManager()

{

if (MainInfoClass.user.RoleId == 3)

{

carAll = carValue.Where(x => x.StatusId == 2 && x.BookingDate > DateTime.Now && x.BookingLast.UserId == MainInfoClass.user.idUser).ToList();

listCarReserv.ItemsSource = carValue.Where(x => x.StatusId == 2 && x.BookingDate > DateTime.Now && x.BookingLast.UserId == MainInfoClass.user.idUser);

}

else

{

listCarReserv.ItemsSource = carValue.Where(x => x.StatusId == 2 && x.BookingDate > DateTime.Now);

}

}

public ICommand RefreshCommand => new Command(async () => await RefreshData());

public async Task RefreshData()

{

try

{

refreshView.IsRefreshing = true;

using (var client = new WebClient())

{

string responze = "";

await Task.Run(() => responze = client.DownloadString(apiBDCars));

carValue = JsonConvert.DeserializeObject<List<Car>>(responze);

var responzeSecond = "";

await Task.Run(() => responzeSecond = client.DownloadString(apiBDBrands));

brandValue = JsonConvert.DeserializeObject<List<Brand>>(responzeSecond);

}

foreach (var item in carValue)

{

item.Model.Brand = brandValue.Where(x => x.idBrand == item.Model.BrandId).FirstOrDefault();

}

carValue = carValue.OrderByDescending(x => x.DateBuy).ToList();

if (loadValue == 1)

{

GetCarAnalit();

}

else if (loadValue == 2)

{

GetCarActivInspect();

}

else if (loadValue == 3)

{

GetCar();

}

else if (loadValue == 4)

{

GetCarManager();

}

else if (loadValue == 5)

{

GetCarHistory();

}

else if (loadValue == 6)

{

GetCarInspect();

}

else if (loadValue == 7)

{

GetCarOther();

}

refreshView.IsRefreshing = false;

}

catch (Exception ex)

{

refreshView.IsRefreshing = false;

DisplayAlert("Внимание", $"Отсутствует подключение к интернету", "ОК");

}

}

private void InsertCarInspector\_Tapped(object sender, EventArgs e)

{

Navigation.PushAsync(new VehicleReportPage(this), false);

}

private async void GoInfoCar\_Tapped(object sender, EventArgs e)

{

await Navigation.PushAsync(new InfoCar((sender as Frame).BindingContext as Car, this), false);

}

private void search\_Focused(object sender, FocusEventArgs e)

{

searchFrame.BorderColor = Color.FromHex("2D4B7F");

searchIcon.Fill = new SolidColorBrush(Color.FromHex("2D4B7F"));

}

private void search\_Unfocused(object sender, FocusEventArgs e)

{

if (searchEntry.Text == null || searchEntry.Text.ToString().Length == 0)

{

searchFrame.BorderColor = Color.LightGray;

searchIcon.Fill = Brush.LightGray;

}

}

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

{

List<Car> cars = new List<Car>();

carAll = carValue.Where(x => x.GosNumber.Contains(searchEntry.Text)).ToList();

carObc = new ObservableRangeCollection<Car>(carAll.Take(ndx));

if (loadValue == 4)

{

listCarReserv.ItemsSource = carObc;

}

else

{

listCar.ItemsSource = carObc;

}

}

private void listCar\_RemainingItemsThresholdReached(object sender, EventArgs e)

{

if (carAll.Count >= ndx)

{

carAll.RemoveRange(0, ndx);

if (carAll.Count >= ndx)

{

carObc.AddRange(carAll.Take(2));

}

else carObc.AddRange(carAll.Take(carAll.Count));

}

}

}

EditPasswordPage.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="CarEvaluation.EditPasswordPage">

<ContentPage.Content>

<ScrollView Orientation="Vertical">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="55"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Frame Grid.Row="0" Padding="0">

<StackLayout VerticalOptions="Center"

HorizontalOptions="FillAndExpand"

Orientation="Horizontal">

<Frame Padding="0" HasShadow="False">

<Path Style="{StaticResource nextIconStyle}"

HorizontalOptions="Start"

Margin="20,0,0,0"

WidthRequest="20"

Rotation="180"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoBack\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Label Text="Редактирование пароля"

HorizontalOptions="CenterAndExpand"

FontSize="18"

Margin="0,0,30,0"

Padding="0"

Style="{StaticResource mainTextStyle}"/>

</StackLayout>

</Frame>

<StackLayout Grid.Row="1" Margin="20,30,20,0" Spacing="10">

<StackLayout Spacing="2">

<Label Text="Старый пароль"

Margin="10,5,0,0"

FontSize="16"/>

<Entry x:Name="oldPassEntry"

IsPassword="True"

TextColor="Black"

TextChanged="oldPassEntry\_TextChanged"

Placeholder="\*\*\*\*\*\*\*"

Margin="6,0,6,0"/>

<Label x:Name="oldPassWarning" Text="Неверный пароль" Style="{StaticResource warningStyle}"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Новый пароль"

Margin="10,5,0,0"

FontSize="16"/>

<Entry x:Name="newPassEntryFirst"

TextColor="Black"

Placeholder="\*\*\*\*\*\*\*"

IsPassword="True"

TextChanged="newPassEntryFirst\_TextChanged"

Margin="6,0,6,0"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Повторите пароль"

Margin="10,5,0,0"

FontSize="16"/>

<Entry x:Name="newPassEntrySecond"

IsPassword="True"

Placeholder="\*\*\*\*\*\*\*"

TextColor="Black"

TextChanged="newPassEntryFirst\_TextChanged"

Margin="6,0,6,0"/>

<Label x:Name="newPassWarning" Text="Пароли не совпадают" Style="{StaticResource warningStyle}"/>

</StackLayout>

<Frame x:Name="Save"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

HeightRequest="50"

Padding="0"

Margin="0,20,0,0">

<Label Text="Сохранить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoSave\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</Grid>

</ScrollView>

</ContentPage.Content>

</ContentPage>

EditPasswordPage.xaml.cs:

public partial class EditPasswordPage : ContentPage

{

public EditPasswordPage()

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

}

private async void GoSave\_Tapped(object sender, EventArgs e)

{

try

{

if (MainInfoClass.user.Password != oldPassEntry.Text)

{

oldPassWarning.IsVisible = true;

}

else if (newPassEntryFirst.Text != newPassEntrySecond.Text)

{

newPassWarning.IsVisible = true;

}

else

{

MainInfoClass.user.Password = newPassEntryFirst.Text;

string z = string.Format("http://crazy-ramanujan.37-140-192-136.plesk.page/api/Users/{0}", MainInfoClass.user.idUser);

using (WebClient wc = new WebClient())

{

wc.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string json = JsonConvert.SerializeObject(MainInfoClass.user);

wc.UploadString(z, "PUT", json);

}

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private async void GoBack\_Tapped(object sender, EventArgs e)

{

await Navigation.PopAsync(false);

}

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

{

oldPassWarning.IsVisible = false;

}

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

{

newPassWarning.IsVisible = false;

}

}

EditProfilePage.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:CarEvaluation"

x:Class="CarEvaluation.EditProfilePage">

<ContentPage.Resources>

<local:IconPersonConverter x:Key="IconPersonConverter"/>

</ContentPage.Resources>

<ContentPage.Content>

<ScrollView Orientation="Vertical">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="55"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Frame Grid.Row="0" Padding="0">

<StackLayout VerticalOptions="Center"

HorizontalOptions="FillAndExpand"

Orientation="Horizontal">

<Frame Padding="0" HasShadow="False">

<Path Style="{StaticResource nextIconStyle}"

HorizontalOptions="Start"

Margin="20,0,0,0"

WidthRequest="20"

Rotation="180"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoBack\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Label Text="Редактирование профиля"

HorizontalOptions="CenterAndExpand"

FontSize="18"

Margin="0,0,30,0"

Padding="0"

Style="{StaticResource mainTextStyle}"/>

</StackLayout>

</Frame>

<StackLayout Grid.Row="1" Margin="20,30,20,0" Spacing="10">

<Grid HorizontalOptions="Center">

<Frame CornerRadius="50"

VerticalOptions="Start"

HorizontalOptions="Center"

Padding="0"

HeightRequest="100"

WidthRequest="100">

<Image x:Name="personIcon" Source="{Binding Photo, Converter={StaticResource IconPersonConverter}}" HeightRequest="90" WidthRequest="90"/>

</Frame>

<Frame BackgroundColor="Gray"

HeightRequest="30"

CornerRadius="50"

Padding="0"

WidthRequest="30"

Margin="70,0,0,0"

VerticalOptions="Start"

HorizontalOptions="Center">

<Path Fill="White"

VerticalOptions="Center"

HorizontalOptions="Center"

Margin="0,0,1,1"

Data="M17.5 15.8333V4.16667C17.5 3.25 16.75 2.5 15.8333 2.5H4.16667C3.25 2.5 2.5 3.25 2.5 4.16667V15.8333C2.5 16.75 3.25 17.5 4.16667 17.5H15.8333C16.75 17.5 17.5 16.75 17.5 15.8333ZM7.08333 11.25L9.16667 13.7583L12.0833 10L15.8333 15H4.16667L7.08333 11.25Z"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoInsertPhoto\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</Grid>

<StackLayout Margin="0,20,0,0" Spacing="2">

<Label Text="Фамилия"

FontSize="16"

Margin="10,5,0,0"/>

<Entry x:Name="surnameEntry"

TextColor="Black"

TextChanged="surnameEntry\_TextChanged"

Margin="6,0,6,0"/>

<Label x:Name="surnameWarning" Text="Заполните поле" Style="{StaticResource warningStyle}"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Имя"

Margin="10,5,0,0"

FontSize="16"/>

<Entry x:Name="nameEntry"

TextColor="Black"

TextChanged="nameEntry\_TextChanged"

Margin="6,0,6,0"/>

<Label x:Name="nameWarning" Text="Заполните поле" Style="{StaticResource warningStyle}"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Отчество"

Margin="10,5,0,0"

FontSize="16"/>

<Entry x:Name="middleNameEntry"

TextColor="Black"

TextChanged="middleNameEntry\_TextChanged"

Margin="6,0,6,0"/>

<Label x:Name="middleNameWarning" Text="Заполните поле" Style="{StaticResource warningStyle}"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Номер телефона"

Margin="10,5,0,0"

FontSize="16"/>

<Entry x:Name="phoneEntry"

TextColor="Black"

Keyboard="Telephone"

MaxLength="12"

TextChanged="phoneEntry\_TextChanged"

Margin="6,0,6,0"/>

<Label x:Name="phoneWarning" Text="Заполните поле" Style="{StaticResource warningStyle}"/>

</StackLayout>

<Frame x:Name="Save"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

HeightRequest="50"

Padding="0"

Margin="0,20,0,0">

<Label Text="Сохранить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoSave\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</Grid>

</ScrollView>

</ContentPage.Content>

</ContentPage>

EditProfilePage.xaml.cs:

public partial class EditProfilePage : ContentPage

{

FileResult photo;

bool isImageDel = false;

public EditProfilePage()

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

surnameEntry.Text = MainInfoClass.user.Surname;

nameEntry.Text = MainInfoClass.user.Name;

middleNameEntry.Text = MainInfoClass.user.MiddleName;

phoneEntry.Text = MainInfoClass.user.Phone;

personIcon.BindingContext = MainInfoClass.user;

if (string.IsNullOrEmpty( MainInfoClass.user.Photo) && photo == null)

{

delFrame.IsVisible = false;

}

}

private async void GoSave\_Tapped(object sender, EventArgs e)

{

try

{

Regex r = new Regex(@"^(\+7)(\d{10})$");

if (String.IsNullOrWhiteSpace(surnameEntry.Text))

{

surnameWarning.IsVisible = true;

}

else if (String.IsNullOrWhiteSpace(nameEntry.Text))

{

nameWarning.IsVisible = true;

}

else if (String.IsNullOrWhiteSpace(middleNameEntry.Text))

{

middleNameWarning.IsVisible = true;

}

else if (String.IsNullOrWhiteSpace(phoneEntry.Text))

{

surnameWarning.IsVisible = true;

}

else if (String.IsNullOrWhiteSpace(phoneEntry.Text))

{

phoneWarning.Text = "Заполните поле";

phoneWarning.IsVisible = true;

}

else if (!r.IsMatch(phoneEntry.Text))

{

phoneWarning.Text = "Неверный формат. Пример: +78888888888";

phoneWarning.IsVisible = true;

}

else if (MainInfoClass.userList.Where(x => x.Phone == phoneEntry.Text).FirstOrDefault() != null)

{

phoneWarning.Text = "Данный номер телефона уже занят";

phoneWarning.IsVisible = true;

}

else

{

MainInfoClass.user.Surname = surnameEntry.Text;

MainInfoClass.user.Name = nameEntry.Text;

MainInfoClass.user.MiddleName = middleNameEntry.Text;

MainInfoClass.user.Phone = phoneEntry.Text;

if (isImageDel)

{

MainInfoClass.user.Photo = null;

}

if (photo != null)

{

string fullApiImage = MainInfoClass.apiImageSource + "/api/ImageUpload";

var restClient = new RestClient(fullApiImage);

restClient.Timeout = -1;

var request = new RestRequest(Method.POST);

request.AddFile("file", photo.FullPath);

IRestResponse response = restClient.Execute(request);

MainInfoClass.user.Photo = MainInfoClass.apiImageSource + response.Content.Replace("\"", "");

}

int indexUser = MainInfoClass.user.idUser;

string z = string.Format("{0}/Users/{1}", MainInfoClass.apiDBSource, indexUser);

using (WebClient wc = new WebClient())

{

User mainUser = (User)MainInfoClass.user.Clone();

wc.Headers.Add(HttpRequestHeader.ContentType, "application/json");

wc.UploadString(z, WebRequestMethods.Http.Put, JsonConvert.SerializeObject(mainUser));

}

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private async void GoBack\_Tapped(object sender, EventArgs e)

{

await Navigation.PopAsync(false);

}

private async void GoInsertPhoto\_Tapped(object sender, EventArgs e)

{

try

{

photo = await MediaPicker.PickPhotoAsync();

if (photo != null)

{

personIcon.Source = photo.FullPath;

delFrame.IsVisible = true;

}

}

catch (Exception ex)

{

DisplayAlert("Внимание", ex.Message, "ОК");

}

}

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

{

phoneWarning.IsVisible = false;

}

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

{

middleNameWarning.IsVisible = false;

}

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

{

nameWarning.IsVisible = false;

}

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

{

surnameWarning.IsVisible = false;

}

private void DeletedImage\_Tapped(object sender, EventArgs e)

{

personIcon.Source = "personDefault.png";

photo = null;

isImageDel = true;

delFrame.IsVisible = false;

}

}

InfoCar.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local ="clr-namespace:CarEvaluation"

x:Class="CarEvaluation.InfoCar">

<ContentPage.Resources>

<local:PTCConverter x:Key="PTCConverter"/>

</ContentPage.Resources>

<ContentPage.Content>

<ScrollView x:Name="Scroll" Orientation="Vertical" Padding="0">

<StackLayout x:Name="stack" Orientation="Vertical">

<Frame Grid.Row="0" Padding="0" HeightRequest="55">

<Grid VerticalOptions="Center"

HorizontalOptions="FillAndExpand">

<Frame Padding="0"

HasShadow="False"

HorizontalOptions="Start"

Margin="10,0,0,0"

WidthRequest="40">

<Path Style="{StaticResource nextIconStyle}"

HorizontalOptions="Center"

Rotation="180"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoBack\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Label Text="Отчёт об авто"

HorizontalOptions="CenterAndExpand"

FontSize="18"

Padding="0"

Style="{StaticResource mainTextStyle}"/>

</Grid>

</Frame>

<StackLayout Margin="30,20,30,0">

<Label

FontSize="22"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1} {2}">

<Binding Path="Model.Brand.Name"/>

<Binding Path="Model.Name"/>

<Binding Path="Generation.Name"/>

</MultiBinding>

</Label.Text>

</Label>

<Label x:Name="reservLabel"

Text="{Binding BookingDate, StringFormat='{}Забронирован до: {0:dd.MM.yy HH\\:mm}'}"

FontSize="15"

Grid.Row="1"

IsVisible="False"

Grid.Column="1"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Frame

BackgroundColor="LightGray"

Padding="0"

CornerRadius="5"

HasShadow="False"

Margin="0,10,0,0"

HeightRequest="200">

<Image x:Name="mainImage" Source="{Binding FirstImage}" />

</Frame>

<CollectionView x:Name="carImages"

HeightRequest="100"

Margin="0"

ItemsSource="{Binding PhotoObserCollection}"

HorizontalOptions="Fill">

<CollectionView.ItemsLayout>

<LinearItemsLayout Orientation="Horizontal"/>

</CollectionView.ItemsLayout>

<CollectionView.ItemTemplate>

<DataTemplate>

<StackLayout Orientation="Horizontal">

<Image Source="{Binding Picture}" HeightRequest="78" >

<Image.GestureRecognizers>

<TapGestureRecognizer Tapped="TapGestureRecognizer\_Tapped"/>

</Image.GestureRecognizers>

</Image>

<Grid WidthRequest="10" />

</StackLayout>

</DataTemplate>

</CollectionView.ItemTemplate>

</CollectionView>

<Label Text="Общая информация"

FontSize="20"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="130"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Text="Статус"

FontSize="15"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="{Binding Status.Name}"

FontSize="15"

HorizontalOptions="Start"

Grid.Column="1"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="Дата осмотра"

FontSize="15"

Grid.Row="1"

HorizontalOptions = "Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="{Binding DateBuy, StringFormat='{}{0:dd.MM.yy}'}"

FontSize="15"

Grid.Row="1"

Grid.Column="1"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="Цена выкупа"

FontSize="15"

Grid.Row="2"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="{Binding IsBuyPrice}"

FontSize="15"

Grid.Row="2"

Grid.Column="1"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="Цена"

FontSize="15"

Grid.Row="3"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="{Binding IsSellPrice}"

FontSize="15"

HorizontalOptions="Start"

Grid.Row="3"

Grid.Column="1"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="Осмотр провёл"

Grid.Row="4"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label HorizontalOptions="Start"

Grid.Row="4"

Grid.Column="1"

Style="{StaticResource labelInfoEmplStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1}">

<Binding Path="user.Surname"/>

<Binding Path="user.Name"/>

</MultiBinding>

</Label.Text>

</Label>

<Label x:Name="analitLabelFirst"

Text="Оценку провёл"

Grid.Row="5"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label x:Name="analitLabelSecond"

HorizontalOptions="Start"

Grid.Row="5"

Grid.Column="1"

Style="{StaticResource labelInfoEmplStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1}">

<Binding Path="user1.Surname"/>

<Binding Path="user1.Name"/>

</MultiBinding>

</Label.Text>

</Label>

</Grid>

<Label Text="Характеристики"

FontSize="20"

Margin="0,10,0,0"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<Grid Margin="0,10,0,0">

<Grid.Resources>

<ResourceDictionary>

<Style x:Key="Info" TargetType="Label">

<Setter Property="TextColor" Value="Gray"/>

<Setter Property="FontSize" Value="15"/>

<Setter Property="FontFamily" Value="NunitoRegular"/>

</Style>

<Style x:Key="InfoFiled" TargetType="Label" BasedOn="{StaticResource Info}">

<Setter Property="TextColor" Value="Black"/>

</Style>

</ResourceDictionary>

</Grid.Resources>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="130"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Text="Год выпуска"

Style="{DynamicResource Info}"

Grid.Column="0"/>

<Label Text="{Binding YearOfRealease}"

Style="{DynamicResource InfoFiled}"

Grid.Column="1"/>

<Label Text="Пробег"

Style="{DynamicResource Info}"

Grid.Row="1"

Grid.Column="0"/>

<Label Text="{Binding Mileage, StringFormat='{}{0} км'}"

Style="{DynamicResource InfoFiled}"

Grid.Row="1"

Grid.Column="1"/>

<Label Text="Кузов"

Style="{DynamicResource Info}"

Grid.Row="2"

Grid.Column="0"/>

<Label Text="{Binding Body.Name}"

Style="{DynamicResource InfoFiled}"

Grid.Row="2"

Grid.Column="1"/>

<Label Text="Цвет"

Style="{DynamicResource Info}"

Grid.Row="3"

Grid.Column="0"/>

<Label Text="{Binding Color.Name}"

Style="{DynamicResource InfoFiled}"

Grid.Row="3"

Grid.Column="1"/>

<Label Text="Двигатель"

Style="{DynamicResource Info}"

Grid.Row="4"

Grid.Column="0"/>

<Label

Style="{DynamicResource InfoFiled}"

Grid.Row="4"

Grid.Column="1">

<Label.Text>

<MultiBinding StringFormat="{}{0} л/{1} л.с./{2}/{3}">

<Binding Path="EngineCapacity"/>

<Binding Path="PowerCapacity"/>

<Binding Path="FuelType.Name"/>

<Binding Path="TypeEngine.Name"/>

</MultiBinding>

</Label.Text>

</Label>

<Label Text="КПП"

Style="{DynamicResource Info}"

Grid.Row="5"

Grid.Column="0"/>

<Label Text="{Binding TypeTransmission.Name}"

Style="{DynamicResource InfoFiled}"

Grid.Row="5"

Grid.Column="1"/>

<Label Text="Привод"

Style="{DynamicResource Info}"

Grid.Row="6"

Grid.Column="0"/>

<Label Text="{Binding Drive.Name}"

Style="{DynamicResource InfoFiled}"

Grid.Row="6"

Grid.Column="1"/>

<Label Text="Владельцы"

Style="{DynamicResource Info}"

Grid.Row="7"

Grid.Column="0"/>

<Label Text="{Binding CountOwner}"

Style="{DynamicResource InfoFiled}"

Grid.Row="7"

Grid.Column="1"/>

<Label Text="Комплектация"

Style="{DynamicResource Info}"

Grid.Row="8"

Grid.Column="0"/>

<Label Text="{Binding Equipment.Name}"

Style="{DynamicResource InfoFiled}"

Grid.Row="8"

Grid.Column="1"/>

<Label Text="ПТС"

Style="{DynamicResource Info}"

Grid.Row="9"

Grid.Column="0"/>

<Label Text="{Binding IsOriginalPTC, Converter= {StaticResource PTCConverter}}"

Style="{DynamicResource InfoFiled}"

Grid.Row="9"

Grid.Column="1"/>

<Label Text="VIN"

Style="{DynamicResource Info}"

Grid.Row="10"

Grid.Column="0"/>

<Label Text="{Binding VIN}"

Style="{DynamicResource InfoFiled}"

Grid.Row="10"

Grid.Column="1"/>

<Label Text="Госномер"

Style="{DynamicResource Info}"

Grid.Row="11"

Grid.Column="0"/>

<Label Text="{Binding GosNumber}"

Style="{DynamicResource InfoFiled}"

Grid.Row="11"

Grid.Column="1"/>

</Grid>

<Label Text="Юридические ограничения"

FontSize="20"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<StackLayout Orientation="Horizontal">

<BoxView x:Name="boxJuridic"

BackgroundColor="Green"

VerticalOptions="Start"

HorizontalOptions="Start"

HeightRequest="8"

Margin="0,7,0,0"

WidthRequest="8"

CornerRadius="50"/>

<Label x:Name="labelJuridic" Text="Отсутствуют"

FontSize="15"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

</StackLayout>

<Label Text="Ремонты по ЛКП"

FontSize="20"

Margin="0,10,0,0"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<StackLayout x:Name="stackRepair">

</StackLayout>

<Label Text="Компрессия мотора"

FontSize="20"

Margin="0,10,0,0"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<Label Text="{Binding CompressionMotor, StringFormat = '{}Результат замера: {0}'}"

FontSize="15"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label Text="Техническое состояние автомобиля"

FontSize="20"

VerticalOptions="Start"

Margin="0,10,0,0"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<Label Text="{Binding Description, StringFormat='{}Описание: {0 }'}"

FontSize="15"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<StackLayout x:Name="infoBooking"

IsVisible="False"

Orientation="Vertical">

<Label Text="Информация о брони"

FontSize="20"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource mainTextStyle}"/>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="130"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Label Text="Клиент"

Grid.Row="0"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label HorizontalOptions="Start"

Grid.Row="0"

Grid.Column="1"

Style="{StaticResource labelInfoEmplStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1}">

<Binding Path="BookingLast.SurnameClient"/>

<Binding Path="BookingLast.NameClient"/>

</MultiBinding>

</Label.Text>

</Label>

<Label Text="Дата начала"

Grid.Row="1"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label HorizontalOptions="Start"

Grid.Row="1"

Grid.Column="1"

Text="{Binding BookingLast.Date, StringFormat='{}{0:dd.MM.yy HH\\:mm}'}"

Style="{StaticResource labelInfoEmplStyle}">

</Label>

<Label Text="Дата конец"

Grid.Row="2"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label HorizontalOptions="Start"

Grid.Row="2"

Grid.Column="1"

Text="{Binding BookingDate, StringFormat='{}{0:dd.MM.yy HH\\:mm}'}"

Style="{StaticResource labelInfoEmplStyle}">

</Label>

<Label Text="Номер телефона"

Grid.Row="3"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label HorizontalOptions="Start"

Grid.Row="3"

Grid.Column="1"

Text="{Binding BookingLast.Phone}"

Style="{StaticResource labelInfoEmplStyle}">

</Label>

<Label Text="Менеджер"

Grid.Row="4"

VerticalOptions="Start"

HorizontalOptions="Start"

Style="{StaticResource labelInfoEmplStyle}"/>

<Label HorizontalOptions="Start"

Grid.Row="4"

Grid.Column="1"

Style="{StaticResource labelInfoEmplStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1}">

<Binding Path="BookingLast.User.Surname"/>

<Binding Path="BookingLast.User.Name"/>

</MultiBinding>

</Label.Text>

</Label>

</Grid>

</StackLayout>

</StackLayout>

<Grid Margin="20,20,20,50">

<StackLayout x:Name="addPrice" IsVisible="False" Orientation="Vertical">

<Frame

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

BackgroundColor="Green"

VerticalOptions="EndAndExpand">

<Label x:Name="doneButton" Text="Одобрить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Done\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Frame

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

BackgroundColor="Red"

VerticalOptions="EndAndExpand">

<Label Text="Отклонить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Cancle\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

<StackLayout x:Name="edit" IsVisible="False" Orientation="Vertical">

<Frame

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

VerticalOptions="EndAndExpand">

<Label Text="Редактировать" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoEdit\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Frame

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

VerticalOptions="EndAndExpand">

<Label Text="Удалить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Deleted\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

<StackLayout x:Name="managStack" IsVisible="false" Margin="0,0,0,0" Orientation="Vertical">

<Frame x:Name="callPhone"

Style="{StaticResource buttonBaseStyle}"

IsVisible="False"

HorizontalOptions="FillAndExpand"

VerticalOptions="EndAndExpand">

<Label Text="Позвонить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="CallPhone\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Grid>

<Frame x:Name="reserve"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

IsVisible="False"

VerticalOptions="EndAndExpand">

<Label Text="Забронировать" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Reserve\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Frame x:Name="reserveClose"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

IsVisible="False"

VerticalOptions="EndAndExpand">

<Label Text="Отменить бронь" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="CloseRezer\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</Grid>

<Frame x:Name="sellClick"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

VerticalOptions="EndAndExpand">

<Label Text="Продан" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Sell\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</Grid>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

InfoCar.xaml.cs:

public partial class InfoCar : ContentPage

{

Car carValue;

CarInspector ci;

public InfoCar(Car car, CarInspector ci)

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

carValue = car;

this.ci = ci;

}

protected override void OnAppearing()

{

base.OnAppearing();

try

{

BindingContext = null;

BindingContext = carValue;

if (carValue.BookingDate > DateTime.Now)

{

reservLabel.IsVisible = true;

infoBooking.IsVisible = true;

}

if (MainInfoClass.user.RoleId == 1 && carValue.StatusId == 1)

{

addPrice.IsVisible = true;

}

else if (MainInfoClass.user.RoleId == 1 && carValue.StatusId == 2 && carValue.BookingDate < DateTime.Now)

{

addPrice.IsVisible = true;

doneButton.Text = "Редактировать";

}

else if (MainInfoClass.user.RoleId == 2 && carValue.StatusId == 1)

{

edit.IsVisible = true;

}

else if (MainInfoClass.user.RoleId == 3 && carValue.BookingDate < DateTime.Now)

{

managStack.IsVisible = true;

reserve.IsVisible = true;

}

else if (MainInfoClass.user.RoleId == 3 && carValue.BookingDate > DateTime.Now)

{

managStack.IsVisible = true;

callPhone.IsVisible = true;

reserveClose.IsVisible = true;

}

if (!carValue.Juridical)

{

boxJuridic.BackgroundColor = Color.Red;

labelJuridic.Text = "Автомобиль юр. не чист";

}

stackRepair.Children.Clear();

if (carValue.Repairedparts.Count == 0)

{

stackRepair.Children.Add(new StackLayout

{

VerticalOptions = LayoutOptions.Start,

Orientation = StackOrientation.Horizontal,

Children =

{

new BoxView

{

BackgroundColor = Color.Green,

VerticalOptions = LayoutOptions.Start,

HorizontalOptions = LayoutOptions.Start,

HeightRequest = 8,

Margin = new Thickness(0,7,0,0),

WidthRequest = 8,

CornerRadius = 50

},

new Label

{

FontSize = 15,

VerticalOptions = LayoutOptions.Start,

HorizontalOptions = LayoutOptions.Start,

TextColor = Color.Black,

FontFamily = "NunitoRegular",

Text = "Отсутствуют"

}

}

});

}

else

{

foreach (var item in carValue.Repairedparts)

{

stackRepair.Children.Add(new StackLayout

{

VerticalOptions = LayoutOptions.Start,

Orientation = StackOrientation.Horizontal,

Children =

{

new BoxView

{

BackgroundColor = Color.Red,

VerticalOptions = LayoutOptions.Start,

HorizontalOptions = LayoutOptions.Start,

HeightRequest = 8,

Margin = new Thickness(0,7,0,0),

WidthRequest = 8,

CornerRadius = 50

},

new StackLayout

{

Orientation = StackOrientation.Vertical,

Children =

{

new Label

{

FontSize = 15,

VerticalOptions = LayoutOptions.Start,

TextColor = Color.Black,

HorizontalOptions = LayoutOptions.Start,

FontFamily = "NunitoRegular",

Text = string.Format("Деталь: {0}",item.Detail.Name)

},

new Label

{

FontSize = 15,

VerticalOptions = LayoutOptions.Start,

TextColor = Color.Black,

HorizontalOptions = LayoutOptions.Start,

FontFamily = "NunitoRegular",

Text = string.Format("Описание: {0}",item.Description)

}

}

}

}

});

}

}

if (carValue.AniliticId == null)

{

analitLabelFirst.IsVisible = false;

analitLabelSecond.IsVisible = false;

}

else

{

analitLabelFirst.IsVisible = true;

analitLabelSecond.IsVisible = true;

}

}

catch (Exception ex)

{

DisplayAlert("Внимание", ex.Message, "ОК");

}

}

private async void GoBack\_Tapped(object sender, EventArgs e)

{

await Navigation.PopAsync(false);

}

private async void GoEdit\_Tapped(object sender, EventArgs e)

{

try

{

await Navigation.PushAsync(new VehicleReportPage(carValue, ci), false);

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private async void Reserve\_Tapped(object sender, EventArgs e)

{

await Navigation.PushAsync(new RezervePage(carValue), false);

}

private async void Cancle\_Tapped(object sender, EventArgs e)

{

try

{

if (await App.Current.MainPage.DisplayAlert("Внимание", $"Вы уверены, что хотите отклонить автомобиле", "Нет", "Да") == false)

{

carValue.StatusId = 3;

carValue.Status = null;

int indexCar = carValue.idCar;

string z = string.Format("{0}/Cars/{1}", MainInfoClass.apiDBSource, indexCar);

using (WebClient wc = new WebClient())

{

wc.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string json = JsonConvert.SerializeObject(carValue);

wc.UploadString(z, WebRequestMethods.Http.Put, json);

}

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private async void Done\_Tapped(object sender, EventArgs e)

{

await Navigation.PushAsync(new CarApprovalPage(BindingContext as Car), false);

}

private async void CloseRezer\_Tapped(object sender, EventArgs e)

{

try

{

if (await App.Current.MainPage.DisplayAlert("Внимание", $"Вы уверены, что хотите отменить бронь", "Нет", "Да") == false)

{

int indx = carValue.Booking.Last().IdBooking;

string z = string.Format("{0}/Bookings/{1}", MainInfoClass.apiDBSource, indx);

using (WebClient client = new WebClient())

{

client.UploadString(z, "DELETE", "");

}

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private async void Sell\_Tapped(object sender, EventArgs e)

{

try

{

carValue.StatusId = 4;

carValue.Status = null;

int indexCar = carValue.idCar;

string z = string.Format("{0}/Cars/{1}", MainInfoClass.apiDBSource, indexCar);

using (WebClient wc = new WebClient())

{

wc.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string json = JsonConvert.SerializeObject(carValue);

wc.UploadString(z, WebRequestMethods.Http.Put, json);

}

await Navigation.PopAsync(false);

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

private void TapGestureRecognizer\_Tapped(object sender, EventArgs e)

{

mainImage.Source = (sender as Image).Source;

}

private async void CallPhone\_Tapped(object sender, EventArgs e)

{

try

{

var phoneDialer = CrossMessaging.Current.PhoneDialer;

if (phoneDialer.CanMakePhoneCall)

phoneDialer.MakePhoneCall(carValue.BookingLast.Phone);

}

catch (Exception ex)

{

await DisplayAlert("Внимание", ex.Message, "ОК");

}

}

private async void Deleted\_Tapped(object sender, EventArgs e)

{

try

{

if (await App.Current.MainPage.DisplayAlert("Внимание", $"Вы уверены, что хотите удалить информацию об автомобиле", "Нет", "Да") == false)

{

carValue.StatusId = 3;

carValue.Status = null;

using (WebClient client = new WebClient())

{

string json = JsonConvert.SerializeObject(carValue);

client.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string z = string.Format("{0}/Cars/{1}", MainInfoClass.apiDBSource, carValue.idCar);

client.UploadString(z, WebRequestMethods.Http.Put, json);

}

(ci as IRefreshListCar).RefreshData();

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

}

InspectorPage.xaml:

<TabbedPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:android="clr-namespace:Xamarin.Forms.PlatformConfiguration.AndroidSpecific;assembly=Xamarin.Forms.Core"

android:TabbedPage.ToolbarPlacement="Bottom"

xmlns:mp="clr-namespace:CarEvaluation"

BarBackgroundColor="White"

SelectedTabColor="#2D4B7F"

UnselectedTabColor="Gray"

x:Class="CarEvaluation.InspectorPage">

<TabbedPage.Children>

</TabbedPage.Children>

</TabbedPage>

PersonPage.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:CarEvaluation"

x:Class="CarEvaluation.PersonPage">

<ContentPage.Resources>

<local:IconPersonConverter x:Key="IconPersonConverter"/>

</ContentPage.Resources>

<ContentPage.Content>

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="55"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Frame Grid.Row="0" Padding="0">

<Label Text="Мой профиль"

HorizontalOptions="Center"

VerticalOptions="Center"

FontSize="18"

Padding="0"

TextColor="Black"

FontFamily="NunitoBold"/>

</Frame>

<Frame VerticalOptions="Center"

Margin="0,0,10,0"

Padding="0"

Grid.Row="0"

WidthRequest="30"

HorizontalOptions="End">

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoExit\_Tapped"/>

</Frame.GestureRecognizers>

<Path Fill="Black" Data="M19.8333 9.33333L18.1883 10.9783L20.0317 12.8333H10.5V15.1667H20.0317L18.1883 17.01L19.8333 18.6667L24.5 14L19.8333 9.33333ZM5.83333 5.83333H14V3.5H5.83333C4.55 3.5 3.5 4.55 3.5 5.83333V22.1667C3.5 23.45 4.55 24.5 5.83333 24.5H14V22.1667H5.83333V5.83333"/>

</Frame>

<Grid Grid.Row="1" Margin="0,-5,0,0" VerticalOptions="FillAndExpand">

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="100"/>

<RowDefinition Height="60"/>

</Grid.RowDefinitions>

<Frame BackgroundColor="Black" Grid.Row="0" Grid.RowSpan="2"/>

<Frame BackgroundColor="White" Margin="10,50,10,10" CornerRadius="40" Grid.Row="1" Grid.RowSpan="4">

</Frame>

<Frame CornerRadius="50" Grid.Row="1" Grid.RowSpan="2"

VerticalOptions="Start" HorizontalOptions="Center"

BackgroundColor="White"

Padding="2"

HeightRequest="101"

WidthRequest="101">

<Frame HeightRequest="80"

Padding="0"

CornerRadius="50"

HorizontalOptions="FillAndExpand"

WidthRequest="80">

<Image Source="{Binding Photo, Converter={StaticResource IconPersonConverter}}" HeightRequest="90" WidthRequest="90"/>

</Frame>

</Frame>

<StackLayout Orientation="Vertical" Grid.Row="2" Spacing="1" Grid.RowSpan="2">

<Label Style="{StaticResource labelInfoEmplStyle}">

<Label.Text>

<MultiBinding StringFormat="{}{0} {1}">

<Binding Path="Surname" />

<Binding Path="Name" />

</MultiBinding>

</Label.Text>

</Label>

<Label Text="{Binding Role.Name}" TextColor="Gray" Style="{StaticResource labelInfoEmplStyle}"/>

</StackLayout>

<StackLayout Grid.Row="3" VerticalOptions="Start" Spacing="0.1" Margin="10,0,10,0">

<Frame HasShadow="False" >

<StackLayout Orientation="Horizontal">

<Label Text="Редактировать профиль" Style="{StaticResource labelInfoEmplStyle}" />

<Path Style="{StaticResource nextIconStyle}"/>

</StackLayout>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoEditProfile\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Line AnchorX="0" AnchorY="100"/>

<Frame HasShadow="False">

<StackLayout Orientation="Horizontal" >

<Label Text="Сменить пароль" Style="{StaticResource labelInfoEmplStyle}"/>

<Path Style="{StaticResource nextIconStyle}"/>

</StackLayout>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoEditPassword\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</Grid>

</Grid>

</ContentPage.Content>

</ContentPage>

InspectorPage.xaml.cs:

public partial class InspectorPage : TabbedPage

{

List<Car> cars = new List<Car>();

public InspectorPage()

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

MainInfoClass.iPage = this;

if (MainInfoClass.user != null)

{

if (MainInfoClass.user.RoleId == 1)

{

Children.Add(new CarInspector(true)

{

Title = "Ожидают оценку",

IconImageSource = "carM.png",

});

Children.Add(new CarInspector("",true)

{

Title = "История",

IconImageSource = "car.png",

});

}

else if (MainInfoClass.user.RoleId == 2)

{

Children.Add(new CarInspector(true, 2)

{

Title = "Активные",

IconImageSource = "carSearch.png",

});

Children.Add(new CarInspector(2, true)

{

Title = "История",

IconImageSource = "car.png",

});

}

else if(MainInfoClass.user.RoleId == 3)

{

Children.Add(new CarInspector(2)

{

Title = "В наличии",

IconImageSource = "car.png",

});

Children.Add(new CarInspector("")

{

Title = "Забронированные",

IconImageSource = "carReserve.png",

});

}

else if (MainInfoClass.user.RoleId == 4)

{

Children.Add(new CarInspector(true)

{

Title = "Ожидают",

IconImageSource = "carM.png",

});

Children.Add(new CarInspector(2)

{

Title = "В наличии",

IconImageSource = "car.png",

});

Children.Add(new CarInspector("")

{

Title = "Зарезерв.",

IconImageSource = "carReserve.png",

});

Children.Add(new CarInspector("",2)

{

Title = "Прочее",

IconImageSource = "car.png",

});

}

Children.Add(new PersonPage

{

Title = "Профиль",

IconImageSource = "person.png",

});

}

}

}

RezervePage.xaml:

<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="CarEvaluation.RezervePage">

<ContentPage.Content>

<ScrollView Orientation="Vertical" Grid.Row="0" Grid.RowSpan="2">

<StackLayout Orientation="Vertical" Grid.Column="0" Margin="20,0,20,0" Grid.ColumnSpan="2">

<Frame Grid.Row="0" Padding="0" HeightRequest="55">

<StackLayout VerticalOptions="Center"

HorizontalOptions="FillAndExpand"

Orientation="Horizontal">

<Frame Padding="0" HasShadow="False">

<Path Style="{StaticResource nextIconStyle}"

HorizontalOptions="Start"

Margin="20,0,0,0"

WidthRequest="20"

Rotation="180"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="GoBack\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

<Label Text="Бронирование"

HorizontalOptions="CenterAndExpand"

FontSize="18"

Margin="0,0,30,0"

Padding="0"

Style="{StaticResource mainTextStyle}"/>

</StackLayout>

</Frame>

<StackLayout Spacing="2">

<Label Text="Фамилия"

Margin="10,5,0,0"

Style="{DynamicResource infoFieldStyle}"/>

<Entry x:Name="surnameEntry"

Style="{StaticResource entryBaseStyle}"

Placeholder="Введите фамилию"

Keyboard="Chat"

TextChanged="surnameEntry\_TextChanged"

MaxLength="17"

TextColor="Black"

Margin="6,0,6,0"/>

<Label x:Name="surnameWarning"

Text="Фамилия не указана"

Style="{StaticResource warningStyle}"/>

</StackLayout>

<StackLayout Spacing="2">

<Label Text="Имя"

Margin="10,5,0,0"

Style="{DynamicResource infoFieldStyle}"/>

<Entry x:Name="nameEntry"

Style="{StaticResource entryBaseStyle}"

Placeholder="Введите имя"

Keyboard="Text"

TextChanged="nameEntry\_TextChanged"

MaxLength="17"

TextColor="Black"

Margin="6,0,6,0"/>

<Label x:Name="nameWarning"

Text="Имя не указано"

Style="{StaticResource warningStyle}"/>

</StackLayout>

<Label Text="Номер телефона"

Margin="10,5,0,0"

Style="{DynamicResource infoFieldStyle}"/>

<Entry x:Name="phoneEntry"

Style="{StaticResource entryBaseStyle}"

Placeholder="Введите номер телефона"

TextChanged="phoneEntry\_TextChanged"

MaxLength="17"

TextColor="Black"

Margin="6,0,6,0"/>

<Label x:Name="phoneWarning"

Text="Номер телефонв не указан"

Style="{StaticResource warningStyle}"/>

<Frame x:Name="Save"

Style="{StaticResource buttonBaseStyle}"

HorizontalOptions="FillAndExpand"

VerticalOptions="StartAndExpand"

Margin="0,20,0,0">

<Label Text="Сохранить" Style="{StaticResource buttonTextStyle}"/>

<Frame.GestureRecognizers>

<TapGestureRecognizer Tapped="Save\_Tapped"/>

</Frame.GestureRecognizers>

</Frame>

</StackLayout>

</ScrollView>

</ContentPage.Content>

</ContentPage>

RezervePage.xaml.cs:

public partial class RezervePage : ContentPage

{

Car carValue;

public RezervePage(Car car)

{

InitializeComponent();

NavigationPage.SetHasNavigationBar(this, false);

carValue = car;

}

private async void Save\_Tapped(object sender, EventArgs e)

{

try

{

Regex r = new Regex(@"^(\+7)(\d{10})$");

bool war = true;

if (String.IsNullOrWhiteSpace(surnameEntry.Text))

{

surnameWarning.IsVisible = true;

war = false;

}

if (String.IsNullOrWhiteSpace(nameEntry.Text))

{

nameWarning.IsVisible = true;

war = false;

}

if (String.IsNullOrWhiteSpace(phoneEntry.Text))

{

phoneWarning.Text = "Номер телефона не указан";

phoneWarning.IsVisible = true;

war = false;

}

else if (!r.IsMatch(phoneEntry.Text))

{

phoneWarning.Text = "Неверный формат. Пример: +78888888888";

phoneWarning.IsVisible = true;

war = false;

}

if (war)

{

Booking book = new Booking();

book.Date = DateTime.Now;

book.Phone = phoneEntry.Text;

book.SurnameClient = surnameEntry.Text;

book.NameClient = nameEntry.Text;

book.CarId = carValue.idCar;

book.UserId = MainInfoClass.user.idUser;

string z = string.Format("{0}/Bookings", MainInfoClass.apiDBSource);

using (WebClient wc = new WebClient())

{

wc.Headers.Add(HttpRequestHeader.ContentType, "application/json");

string html = wc.UploadString(z, JsonConvert.SerializeObject(book));

carValue.Booking.Add(JsonConvert.DeserializeObject<Booking>(html));

}

(MainInfoClass.carInspector as IRefreshListCar).RefreshData();

await Navigation.PopAsync(false);

}

}

catch (Exception ex)

{

await DisplayAlert("Внимание", "Отсутствует подключение к интернету", "ОК");

}

}

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

{

phoneWarning.IsVisible = false;

}

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

{

nameWarning.IsVisible = false;

}

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

{

surnameWarning.IsVisible = false;

}

private async void GoBack\_Tapped(object sender, EventArgs e)

{

if (await App.Current.MainPage.DisplayAlert("Внимание", "Вы уверены, что хотите отменить бронирование", "Нет", "Да") == false)

{

await Navigation.PopAsync(false);

}

}

}

App.xaml:

<Application xmlns="http://xamarin.com/schemas/2014/forms"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

xmlns:local="clr-namespace:CarEvaluation"

x:Class="CarEvaluation.App">

<Application.Resources>

<ResourceDictionary>

<Color x:Key="BluBiolet">#2a29e7</Color>

<Color x:Key="BluGr">#2D4B7F</Color>

<Style x:Key="baseStyle" TargetType="Grid" >

<Setter Property="Background" Value="White"/>

</Style>

<Style x:Key="buttonBaseStyle" TargetType="Frame" >

<Setter Property="Background" Value="#2D4B7F"/>

<Setter Property="HeightRequest" Value="50"/>

<Setter Property="Padding" Value="0"/>

<Setter Property="HeightRequest" Value="50"/>

<Setter Property="CornerRadius" Value="25"/>

</Style>

<Style x:Key="buttonTextStyle" TargetType="Label" >

<Setter Property="FontFamily" Value="NunitoBold"/>

<Setter Property="TextColor" Value="White"/>

<Setter Property="HorizontalOptions" Value="CenterAndExpand"/>

<Setter Property="VerticalTextAlignment" Value="Center"/>

<Setter Property="FontSize" Value="17"/>

</Style>

<Style x:Key="mainTextStyle" TargetType="Label" >

<Setter Property="FontFamily" Value="NunitoBold"/>

<Setter Property="TextColor" Value="Black"/>

<Setter Property="FontSize" Value="18"/>

</Style>

<Style x:Key="infoCarStyle" TargetType="Label" >

<Setter Property="FontFamily" Value="NunitoRegular"/>

<Setter Property="TextColor" Value="Black"/>

<Setter Property="FontSize" Value="14"/>

</Style>

<Style x:Key="infoFieldStyle" TargetType="Label" >

<Setter Property="FontFamily" Value="NunitoRegular"/>

<Setter Property="TextColor" Value="Gray"/>

<Setter Property="FontSize" Value="16"/>

</Style>

<Style x:Key="labelBaseStyle" TargetType="Label">

<Setter Property="FontFamily" Value="NunitoExtraBold"/>

<Setter Property="TextColor" Value="Black"/>

<Setter Property="HorizontalOptions" Value="Center"/>

<Setter Property="FontSize" Value="37"/>

</Style>

<Style x:Key="labelInfoEmplStyle" TargetType="Label">

<Setter Property="FontFamily" Value="NunitoRegular"/>

<Setter Property="TextColor" Value="#181513"/>

<Setter Property="HorizontalOptions" Value="Center"/>

<Setter Property="FontSize" Value="15"/>

</Style>

<Style x:Key="nextIconStyle" TargetType="Path">

<Setter Property="Data" Value="M6.22998 20.23L7.99998 22L18 12L7.99998 2L6.22998 3.77L14.46 12L6.22998 20.23Z"/>

<Setter Property="Fill" Value="#181513"/>

<Setter Property="HorizontalOptions" Value="EndAndExpand"/>

</Style>

<Style x:Key="entryBaseStyle" TargetType="Entry">

<Setter Property="FontFamily" Value="NunitoRegular"/>

<Setter Property="TextColor" Value="Black"/>

<Setter Property="FontSize" Value="18"/>

</Style>

<Style x:Key="pickerBaseStyle" TargetType="Picker">

<Setter Property="FontFamily" Value="NunitoRegular"/>

<Setter Property="TextColor" Value="Black"/>

<Setter Property="FontSize" Value="18"/>

</Style>

<Style x:Key="warningStyle" TargetType="Label">

<Setter Property="TextColor" Value="Red"/>

<Setter Property="FontSize" Value="13"/>

<Setter Property="IsVisible" Value="False"/>

<Setter Property="Margin" Value="10,-10,0,0"/>

<Setter Property="FontFamily" Value="NunutoMedium"/>

</Style>

</ResourceDictionary>

</Application.Resources>

</Application>

public partial class App : Application

{

public App()

{

InitializeComponent();

MainPage = new NavigationPage(new Authorization());

}

protected override void OnStart()

{

}

protected override void OnSleep()

{

}

protected override void OnResume()

{

}

}

class IconPersonConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if(value == null)

{

return "personDefault.png";

}

else

{

return value;

}

}

public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

class PTCConverter : IValueConverter

{

public object Convert(object value, Type targetType, object parameter, CultureInfo culture)

{

if ((bool)value == true)

{

return "Оригинал";

}

else

{

return "Дубликат";

}

}