МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Бортовой журнал автомобиля»

Выполнил студент Филиппов Виталий Леонидович

(Ф.И.О.)

Руководитель проекта ассистент Колмаков М.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Заведующий кафедрой к.т.н., доц. Смелов В.В .

(учен. степень, звание, должность, Ф.И.О., подпись)

Консультант: ассистент Колмаков М.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Нормоконтролер: ассистент Колмаков М.В.

(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовой проект защищен с оценкой

Минск 2020

Содержание

[**Введение** 3](#_Toc43040249)

[**1. Обзор аналогов приложения «Бортовой журнал автомобиля»** 5](#_Toc43040250)

[**1.1 Приложение «Моя машина»** 5](#_Toc43040251)

[**1.2 Приложение «Car Expenses»** 6](#_Toc43040252)

[**1.3 Приложение «aCar»** 7](#_Toc43040253)

[**1.4 Выводы по главе** 8](#_Toc43040254)

[**2. Проектирование программного продукта** 10](#_Toc43040255)

[**2.1 Проектирование мобильного приложения** 10](#_Toc43040256)

[**2.2 Проектирование сервера** 12](#_Toc43040257)

[**2.3 Проектирование базы данных** 13](#_Toc43040258)

[**2.4 Выводы по главе «Проектирование программного продукта»** 16](#_Toc43040259)

[**3. Реализация приложения** 17](#_Toc43040260)

[**3.1 Технические средства для реализации приложения** 17](#_Toc43040261)

[**3.2 Разработка баз данных** 19](#_Toc43040262)

[**3.3 Разработка клиентской части приложения** 20](#_Toc43040263)

[**3.4 Разработка серверной части приложения** 23](#_Toc43040264)

[**3.5 Выводы по главе «Программная реализация приложений»** 26](#_Toc43040265)

[**4 Руководство пользователя** 27](#_Toc43040266)

[**4.1 Выводы по главе «Руководство пользователя»** 35](#_Toc43040267)

[**ЗАКЛЮЧЕНИЕ** 36](#_Toc43040268)

[**Список использованных источников** 37](#_Toc43040269)

## **Введение**

История средств передвижения уходит корнями в далёкое прошлое, когда человек приручил лошадь, поняв, что на ней он сможет преодолевать большие расстояния. Жажда познания мира, стремление увидеть неизведанные дали заставили людей искать способы и средства более быстрого перемещения по Земле. Конец 19 века подарил людям автомобиль, и вот уже больше ста лет это транспортное средство не теряет своей популярности. Первым, кто изобрёл автомобиль, был Карл Бенц. В 1886 году произошла первая регистрация автомобиля как транспортного средства. Он получил широкое общественное признание, результатом которого стало промышленное производство таких машин.[1]

В современном мире, имея возможность использовать компьютерные разработки, появилась возможность облегчить с их помощью свою жизнь. И жизнь авто в том числе. Для этого не обязательно покупать узкоспециализированное оборудование вроде регистратора или навигатора, или вести блокнот с информацией о транспортном средстве, достаточно лишь установить на мобильный телефон соответствующую программу. Мобильные приложения могут заметно облегчить жизнь и даже сэкономить время и деньги.

Среди автовладельцев можно найти немало тех, кто привык вести скрупулёзный учёт всех затрат, расходов и событий, происходящих с автомобилем во время его эксплуатации. Специально для такой категории водителей существуют мобильные приложения, представляющее собой аналог бортового журнала. Программы позволяет записывать и хранить историю транспортного средства, вести контроль расхода топлива, получать напоминания о предстоящем техническом обслуживании и многое другое.

Целью моего курсового проекта является создание приложения для учёта затрат и расходов на обслуживание автомобиля, а также хранения всех данных о предстоящих событиях, сервисе, характеристиках, пробеге автомобиля в одном месте. Для оптимизации затрат на эксплуатацию автомобиля данные также представлены в виде графиков и диаграмм.

Задачи курсового проекта:

1. Изучить аналогичные программные средства по теме курсового проекта.
2. Спроектировать приложение бортового журнала автомобиля.
3. Реализовать приложение.
4. Протестировать приложение.
5. Разработать руководство пользователя по управлению бортовым журналом.

Актуальность данной работы заключается в том, что в век развития информационных технологий этот вариант будет оптимальным и удобным, так как вся информация о вашем транспортном средстве будет храниться в одном приложении на смартфоне, и пользователь легко сможет им воспользоваться, независимо от его местонахождения.

## **1. Обзор аналогов приложения «Бортовой журнал автомобиля»**

### **1.1 Приложение «Моя машина»**

В приложении представлены возможности добавления своего транспортного средства, заполнения данных о нём, добавление записей в раздел «Сервис», «Заправка», создание напоминаний, а также просмотр статистики. Приложение представлено ниже на скриншотах (Рисунок 1.1).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 1.1 – Приложение «Моя машина»

Приложение предоставляет возможность добавления и хранения данных о вашей машине, а также просмотр статистики о расходах.

### **1.2 Приложение «Car Expenses»**

Главной целью данного приложения является подсчёт расходов на топливо, содержание, и обслуживание автомобиля.

Мобильное приложение представлено на скриншотах ниже (Рисунок 1.2).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 1.2 –Приложение «Car Expenses»

Главными особенностями данного приложения являются расчёт трат, планирование сервисных действий, расход топлива, а также подробная статистика в виде графиков.

### **1.3 Приложение «aCar»**

Данное приложение предназначено для заполнения и хранения данных об обслуживании, услугах, расходах и поездках.

Общее представление о приложении можно получить исходя из скриншотов ниже (Рисунок 1.3).

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 1.3 – Приложение «aCar»

Приложение позволяет управлять набором функций после регистрации и добавления своего транспортного средства, такого как автомобиль, или грузовик.

### **1.4 Выводы по главе**

Были рассмотрены мобильные приложения, выполняющие функции добавления и хранения данных о расходах топлива, техническом обслуживании, содержании автомобиля, статистика затратов, а также сведения о ремонте и поездках. В результате по каждому приложению были выделены положительные моменты, а также недостатки.

Первое приложение называется «Моя Машина». Из положительных моментов следует выделить:

* дизайн в светлых тонах с яркими элементами
* возможность выбрать свою тему оформления
* автоматическое добавление иконок с маркой автомобиля
* автоматическая статистика по расходах
* синхронизация с облачным хранением

Из недостатков данного приложения можно выделить:

* неудобный пользовательский интерфейс
* при выборе другого цвета оформления, некоторые компоненты не сочетаются между собой
* статистика по расходам доступна только за текущий месяц
* количество транспортных средств, которые можно добавить, ограничено
* небольшое количество функций

Следующее из рассмотренных приложений называется «Car Expenses». Из положительных моментов можно выделить:

* интуитивно понятный пользовательский интерфейс
* возможность добавления своей фотографии автомобиля
* большой функционал
* работа в офлайн режиме
* статистика по расходам доступна за любой выбранный пользователем период
* автоматическое сохранение изменений
* возможность добавления своих категорий, если подходящих нет

Из недостатков этого приложения можно выделить:

* сильная перегруженность интерфейса
* не является кроссплатформенное приложением
* неудобная форма заполнения данных о заправке автомобиля
* отсутствие возможности синхронизации данных

Последнее рассмотренное приложение называется «aCar». Из положительных моментов следует отметить:

* добавление фотографий своего транспортного средства
* возможность просмотра статистики в виде графиков
* количество добавленных транспортных средств не ограничено

Из недостатков данного приложения можно отметить следующее:

* устаревший дизайн приложения
* неудобный пользовательский интерфейс
* маленький функционал
* большое количество рекламы, которая перекрывает функции приложения
* названия некоторых элементов на другом языке
* неудобные формы для заполнения данных
* не гармоничная цветовая палитра

Все положительные моменты, а также недостатки были учтены в рамках приложения «Бортовой журнал автомобиля».

## **2. Проектирование программного продукта**

### **2.1 Проектирование мобильного приложения**

В рамках курсового проекта необходимо разработать мобильное приложение, позволяющее пользователю добавлять и редактировать автомобили, предоставлять другому пользователю доступ к автомобилю. Разработать оптимальный алгоритм синхронизации работы пользователей в приложении, так же организовать работу в режиме офлайн. Для автомобиля необходимо реализовать возможность создания и редактирования событий различных типов, предоставления отчётов в виде наглядных графиков и диаграмм.

На главной странице приложение должно позволить пользователю выбрать: просмотреть или добавить событие. Также на этой странице должен быть предоставлен доступ к выбору типа просматриваемых событий.

Необходимо разработать страницу, на которой будет список всех автомобилей, с возможностью выбора активного автомобиля, по которому будет происходить поиск и добавление событий.

Если же пользователь выбирает просмотреть машину, то должна быть отображена страница с краткой информацией о машине, возможностью предоставить доступ другому пользователю и кнопками для перехода к редактированию автомобиля, просмотра отчётов по пробегу и затрат в текущем месяце. Возможность предоставления другому пользователю автомобиля должна быть реализована через оповещения его через почту. Следовательно, должно быть поле для ввода email, на которое будет высылаться сообщение, уведомляющее о том, что пользователю предоставили доступ, и для добавления автомобиля в список своих автомобилей нужно перейти по ссылке.

Если пользователь выбрал просмотр информации о событии, то должна быть предоставлена страница с информацией об этом событии и графиком затрат пользователя на этот тип в текущем месяце. Так же должна существовать возможность перехода на страницу редактирования, и подробного отчёта затрат пользователя в виде графиков и диаграмм.

Среди отчётов в виде графиков и диаграмм требуется отчёт о ежедневном пробеге автомобиля, общем пробеге, а также динамике цен на топливо, затраты по различным категориям и в целом.

Когда пользователь выбирает категорию события, должен быть предоставлен список событий этой категории. Так же должна быть добавлена категория для запланированных событий., где будут находиться события с датой после текущей.

При нажатии на иконку пользователя должен появляться экран с редактированием информации пользователя.

Необходимо реализовать такие функции как: «Забыли пароль», «Регистрацию пользователя», «Синхронизацию с удалённой базой данных»;

Можно ознакомиться с UML-диаграммой, а именно диаграммой вариантов использования. Она отображает функциональность программного продукта с точки зрения получения значимого результата для пользователя. Диаграмма приведена в приложении А.

Таким образом, приложение должно выполнять функции:

* предоставление списка событий;
* добавление автомобиля
* выбор активного автомобиля
* редактирование автомобиля
* предоставление другому пользователю доступа к автомобилю.
* хранение и добавление событий для автомобиля;
* вывод событий по категориям;
* синхронизация данных между пользователями одного автомобиля;
* предоставление наглядных отчётов в виде графиков и диаграмм;

### **2.2 Проектирование сервера**

Для синхронизации двух и более клиентов, а также хранения их информации на удалённом сервере необходимо создать серверное приложение, которое предоставит удобное API для пользования. Так же нужно разработать функционал для возможности синхронизации данных между пользователями одного автомобиля. Передаваемые или получаемые данные от сервера должны передаваться в формате JSON, а также XML.

Серверное приложение должно быть построено на основе архитектурного стиля REST и SOAP. Сервер должен иметь микро сервисную архитектуру, для легкого масштабирования проекта и повторного использования кода. Обращение к базе данных должно быть реализовано через сервисы.

При обращении к серверу по заданному маршруту и определенному HTTP методу сервер будет определять с какой сущностью базы данных необходимо взаимодействовать и какие действия над данными необходимо будет произвести.

При помощи HTTP метода GET сервер должен возвращать запрашиваемые данные клиенту, ссылаясь на аргументы в запросе, как на условие для выборки данных. Метод POST в свою очередь должен позволять передать данные на сервер для последующей обработки. Метод PUT – обновить соответствующие данные, переданные с клиента. Метод DELETE – удалять с базы данных заданную запись.

### **2.3 Проектирование базы данных**

В базе данных данного приложения должно быть реализовано 11 таблиц. Для определённых таблиц должны быть созданы триггеры для обеспечения логирования всех изменений.

В таблице «AUTHENTICATION» содержатся данные для авторизации пользователя. Столбцы данной таблицы:

* Login – символьный тип данных, является первичным ключом;
* idUser – уникальный идентификатор, является внешним ключом на таблицу «Users»;
* Password – символьный тип данных, не должен содержать значений null;
* LastModify – тип данных дата, не должен содержать значений null;
* DisableUser – целочисленный тип данных (bit), не должен содержать значений null;

В таблице «Users» хранится информация о пользователях приложения. Столбцы данной таблицы:

* UserId – уникальный идентификатор, является первичным ключом;
* Name – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* Lastname – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* Patronymic – строковый тип данных, максимальной длинной 30 символов, не должен содержать значений null;
* Address – строковый тип данных, максимальной длинной 60 символов, не должен содержать значений null;
* City – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* Phone – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* Birthday – тип данных дата, не должен содержать значений null;
* Photo – строковый тип данных;
* Sex – символьный тип данных, максимальной длинной 1 символ.

Данная таблица имеет триггер «AuditTrigger\_Users», добавленный на обновление записи.

В таблице «Cars» содержится информация о машинах. Столбцы данной таблицы:

* CarId – уникальный идентификатор, является первичным ключом;
* TypeFuel – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* TypeTransmission – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* Mark – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* Model – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* VolumeEngine – целочисленный тип данных, не должен содержать значений null;
* Power – целочисленный тип данных, не должен содержать значений null;
* Active – целочисленный тип данных (bit), не должен содержать значений null;
* VIN – строковый тип данных, максимальной длинной 17 символов, не должен содержать значений null;
* Comment – строковый тип данных, максимальной длинной 17 символов;
* Photo – строковый тип данных;
* YearIssue – целочисленный тип данных, не должен содержать значений null;

Данная таблица содержит триггер «AuditTrigger\_Cars», добавленный на обновление записи. Ключевой особенностью триггера является то, что он создаёт записи в соответствующей таблице для каждого пользователя, у которого есть эта машина.

В таблице «CarEvents» содержится информация о событиях, связанных с автомобилем. Столбцы данной таблицы:

* EventId – уникальный идентификатор, является первичным ключом;
* idCar – уникальный идентификатор, является внешним ключом;
* idUser – уникальный идентификатор, является внешним ключом;
* idTypeEvents – уникальный идентификатор, является внешним ключом;
* AddressStation – строковый тип данных, максимальной длинной 100 символов;
* Photo – строковый тип данных;
* Comment – строковый тип данных, максимальной длинной 200 символов, не должен содержать значений null;
* Mileage – целочисленный тип данных, не должен содержать значений null;
* Costs – числовой тип данных, не должен содержать значений null;
* UnitPrice – числовой тип данных;
* Date – тип данных дата, не должен содержать значений null;

Данная таблица содержит триггер «AuditTrigger\_CarEvents» на обновление записей таблицы. Ключевой особенностью триггера является то, что он создаёт записи в соответствующей таблице для каждого пользователя, у которого есть эта машина.

В таблице «ActionAudit» содержится информация об изменениях базы данных. Данная таблица заполняется с помощью триггеров. Столбцы данной таблицы:

* Entity – строковый тип данных, максимальной длинной 50 символов, не должен содержать значений null;
* EntityId – уникальный идентификатор, не должен содержать значений null;
* IdUser – уникальный идентификатор, является внешним ключом;
* Action – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null;
* DateUpdate – тип данных дата, не должен содержать значений null;

Данная таблица имеет составной первичный ключ, состоящий из полей: EntityId, DateUpdate, IdUser;

В таблице «UsersCars» хранится информация о id пользователя и id машины для связки таблиц «Users» и «Cars»

* idUser – уникальный идентификатор, является внешним ключом;
* idCar – уникальный идентификатор, является внешним ключом;

Данная таблица имеет составной первичный ключ, состоящий из полей: idUser и idCar. Так же эта таблица должна иметь триггер «AuditTrigger\_UserCars» на добавление и удаление записей.

В таблице «TypeEvents» содержится информация о типах возможных событий. Столбцы данной таблицы:

* TypeEventId – уникальный идентификатор, является первичным ключом;
* TypeName – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null, и содержит только уникальные значения;

В таблице «TypeServices» содержится информация о типах возможных обслуживаний автомобиля. Столбцы данной таблицы:

* TypeServiceId – уникальный идентификатор, является первичным ключом;
* TypeName – строковый тип данных, максимальной длинной 20 символов, не должен содержать значений null, и содержит только уникальные значения;

В таблице «Fuels» содержится информация, дополняющая информацию таблицы «CarEvents» для событий заправок автомобиля. Столбцы данной таблицы:

* FuelId – уникальный идентификатор, является первичным ключом;
* idEvent – уникальный идентификатор, является внешним ключом;
* Volume – числовой тип данных, не должен содержать значений null;

Данная таблица содержит триггер «AuditTrigger\_Fuels» на обновление записей таблицы. Ключевой особенностью триггера является то, что он создаёт записи в соответствующей таблице для каждого пользователя, у которого есть эта машина.

В таблице «CarServices» содержится информация, дополняющая информацию таблицы «CarEvents» для событий обслуживания автомобиля. Столбцы данной таблицы:

* ServiceId – уникальный идентификатор, является первичным ключом;
* idEvent – уникальный идентификатор, является внешним ключом;
* idTypeService – уникальный идентификатор, является внешним ключом;
* Name – строковый тип данных, максимальной длинной 100 символов, не должен содержать значений null, и содержит только уникальные значения;

Данная таблица содержит триггер «AuditTrigger\_CarServices» на обновление записей таблицы. Ключевой особенностью триггера является то, что он создаёт записи в соответствующей таблице для каждого пользователя, у которого есть эта машина.

В таблице «Details» содержится информация добавленных во время обслуживания автомобиля деталей. Столбцы данной таблицы:

* DetailId – уникальный идентификатор, является первичным ключом;
* idCar – уникальный идентификатор, является внешним ключом, и служит для упрощения использования и поддержания базы данных, значение вычисляется с помощью триггера, на основе таблицы «CarEvents»;
* idService – уникальный идентификатор, является внешним ключом;
* Type – строковый тип данных, максимальной длинной 100 символов;
* Name – строковый тип данных, максимальной длинной 100 символов;

Данная таблица содержит триггер «AuditTrigger\_Details» на обновление записей таблицы. Ключевой особенностью триггера является то, что он создаёт записи в соответствующей таблице для каждого пользователя, у которого есть эта машина. А также триггер «InsteadOfInsertDetails», добавленный вместо операции Insert, служит для вычисления поля idCar.

Диаграмма базы данных приведена в приложении Б.

База данных для мобильного устройства должна быть реализована в соответствии с удалённой базой данных, исключая лишь таблицы: «UsersCars», «AUTHENTICATION». База данных должна содержать данные только одного пользователя.

### **2.4 Выводы по главе «Проектирование программного продукта»**

В данной главе было рассмотрено проектирование каждого отдельного компонента приложения.

По части мобильного приложения был определен основной функционал приложения, также были разработаны: диаграмма вариантов использования приложения, алгоритм синхронизации со стороны клиента.

Была также спроектирована база данных, где были определены необходимые таблицы, поля для каждой из них, определены типы данных и ограничения целостности. Также были установлены связи между таблицами, которые описаны на схеме базы данных.

При проектировании серверного приложения был определен архитектурный стиль взаимодействия между клиентом и сервером, формат передаваемых данных, и HTTP методы которые будут использованы в работе с серверной частью. Был представлен алгоритм синхронизации со стороны сервера.

## **3. Реализация приложения**

### **3.1 Технические средства для реализации приложения**

При разработке мобильного приложения использовалась среда разработки Visual Studio Code – это редактор, при этом обладающий функциями IDE, полагающийся на расширения. Таким расширением были установка плагина для поддержки языка Dart и мобильной платформы flutter. Flutter — молодая, но очень многообещающая платформа, уже привлекшая к себе внимание крупных компаний, которые запустили свои приложения. Интересна эта платформа своей простотой сравнимой с разработкой веб-приложений, и скоростью работы наравне с нативными приложениями. Высокая производительность приложения и скорость разработки достигается за счет нескольких техник:

* Не использует JavaScript ни в каком виде;
* Язык программирования Dart, который компилируется в бинарный код;
* Не использует нативные компоненты;
* Для построения UI во flutter используется декларативный подход, на основе виджетов, которые перерисовываются по необходимости. [2]

В качестве сервера выступает ASP.NET Core приложение. ASP.NET Core – это новый веб-фреймворк от Microsoft. Он был переработан с нуля, чтобы быть быстрым, гибким, современным и работать на разных платформах. В дальнейшем ASP.NET Core – это фреймворк, который можно использовать для веб-разработки с .NET. Для работы с базой данных использовался Entity Framework, позволяющий работать с таблицами базы данных как с классами C#, с полями этих таблиц – как со свойствами классов, а синтаксис SQL-запросов заменен на более удобный подход с [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

В качестве удаленной базы данных использовался Mictosoft Sql Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

Для работы с локальной базы данных использовали SQFlite – одна из наиболее часто используемых и актуальных библиотек для подключения SQLite базы данных во Flutter.[3] Это база данных, которая настроена на нуль, что означает, как и другие базы данных, которые вам не нужно настраивать в вашей системе. SQLite не является автономным процессом, как другие базы данных, вы можете связать его статически или динамически в соответствии с вашим требованием с вашим приложением. SQLite напрямую обращается к своим файлам хранения.

При разработке серверной части приложения также применялись следующие технологии:

REST – это стиль архитектуры программного обеспечения для распределенный систем, таких как World Wide Web, который, как правило используется для построения веб-служб. Термин REST был введен в 2000 году Роем Филдингом, одним из авторов HTTP-протокола. Системы, поддерживающие REST, называются RESTful-системами. В общем случае REST является очень простым интерфейсом управления информацией без использования каких-то дополнительных внутренних прослоек. Каждая единица информации однозначно определяется глобальным идентификатором, таким как URL. Каждая URL в свою очередь имеет строго заданный формат. [4]

SOAP – протокол обмена структурированными сообщениями в распределенной вычислительной среде. Первоначально предназначался для реализации удаленного вызова процедур, сейчас используется для обмена сообщениями в формате XML. [5]

REST и SOAP на самом деле не сопоставимы. REST – это архитектурный стиль. SOAP – это формат обмена сообщениями. В SOAP используется формат SOAP XML для запросов и ответов. В REST такого фиксированного формата нет. Возможен обмен сообщениями на основе XML, JSON или любого другого удобного формата. JSON является самым популярным среди используемых форматов.

### **3.2 Разработка баз данных**

Для хранения данных на телефоне используем СУБД SQLite. Доступ к ней осуществляется с помощью библиотеки sqflite. Для сервера будет использована СУБД MSSQL.

Скрипт создания структуры мобильной базы данных представлен в Приложении В, серверной в Приложении Г.

Так же необходимо было реализовать логирования изменения записей таблиц с помощью триггеров. Скрипт создания со стороны удалённой базы данных представлен в приложении Д, со стороны клиентской в приложении E.

Для предоставления отчётов в виде графиков и диаграмм были разработаны соответствующие представления со стороны клиента. Скрипт представлен в приложении Ж.

Мобильная база данных содержит в себе информацию об одном авторизованном пользователе. В случае если в приложение зашёл другой пользователь – таблицы очищаются.

### **3.3 Разработка клиентской части приложения**

Мобильное приложение было реализовано в редакторе –Visual Studio Code. Для написания приложения была использована мобильная платформа Flutter и язык программирования Dart. Платформа позволяет создавать кроссплатформенные приложения. Интерфейс создаётся с помощью виджетов, которые представляют собой компоненты с разметкой, ключевой особенностью является то, что при необходимости виджет может перерисовываться. Что позволяет быстро изменять интерфейс приложения. Единой точкой входа в приложение является виджет MyApp, расположенный в файле main.dart.

Графическая структура проекта представлена на рисунке 3.3.1.

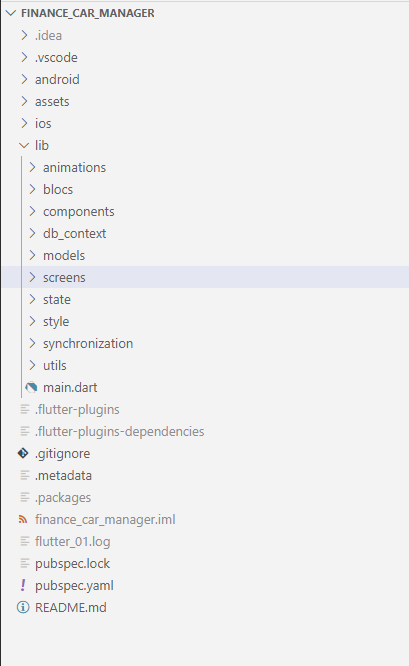


Рисунок 3.3.1 – Графическая структура проекта

Архитектура приложения представлена на рисунке 3.3.2.

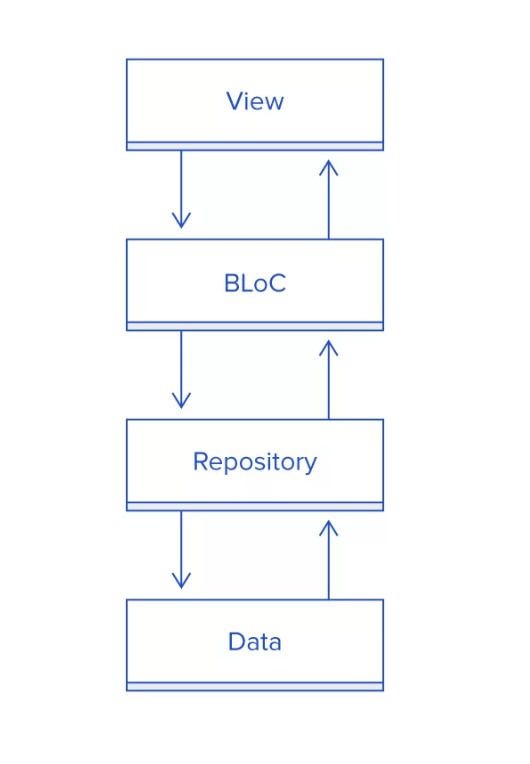


Рисунок 3.3.2 – Архитектура мобильного приложения

В папке db\_context описан слой «Repository», он позволяет получать доступ к слою «Data», доступ осуществляется с помощью вызова нужного метода, для запроса в базу данных. В папке blocs описан слой, который предоставляет механизм, при котором слой «View» подписывается на изменения коллекций, получаемых со слоя «Repository». Таким образом, интерфейс содержит актуальную информацию. Так же верхний слой, отвечающий за интерфейс приложения, может вызывать методы слоя «Bloc» для манипулирования с данными. Папка components содержит все наши виджеты, а папка screens содержит сформированные экраны приложения. В папке models находятся все объекты, отражающие таблицы мобильной базы данных. А также вспомогательные объекты. Так, например класс MyApi представлен на рисунке 3.3.3, содержит все uri сервера и методы отправки запросов на него. Все запросы содержат в заголовке token, который выдаётся сервером при входе в приложение, и храниться в SharedPreferences.



Рисунок 3.3.3 – Класс MyApi

При разработке функционала с добавлением фото, возникли проблемы. Размер фото превосходил размер курсора считывания в Sqlite, чтобы обойти эту проблему была использована NoSql база данных Hive, в которой храниться пары ключ: значение. Возможности этой базы данных во много раз опережают Sqlite, но оптимальной работы приложения с таким решением не получилось, так как файл считывался медленно, и затормаживал интерфейс пользователя. Было принято решение сжимать загружаемые фото. Для сжатия без потери качества использовали файлы формата .png. Тестирование приложения показало, что скорость работы увеличилось, но не до достаточного уровня. Таким образом, функционал добавления фото не был реализован в полной мере.

Для разработки синхронизации с сервером был построен алгоритм, который представлен в приложении З.

### **3.4 Разработка серверной части приложения**

При написании серверной логики было решено разделить сервер на 3 проекта, чтобы получить максимально независимые части, что в дальнейшем предоставляет много плюсов в разработке приложения и поддержании такого кода. Архитектура проекта представлена на рисунке 3.4.1.

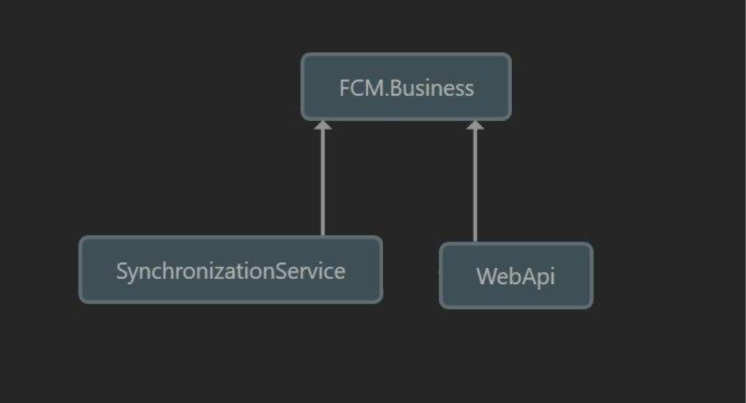


Рисунок 3.4.1 – Архитектура проекта

«FCM.Business» – библиотека классов, предоставляет сервисы для работы с базой данных. Схема классов данного проекта представлена в приложении И. Каждый сервис, условно, предназначен для работы только с одной сущностью, что позволяет выстраивать микро сервисную архитектуру в рамках проекта. В дальнейшем, посредством IoC контейнеров, данные сервисы используются в проектах.

«WebApi» – проект Asp.Net Core, предназначенный для создания удобного Api для работы клиента и сервера.

«SynchronizationService» – сервер, который предназначен для выдачи клиенту контракта синхронизации с помощью протокола SOAP. Алгоритм работы представлен в приложении К.

Данные проекты были реализованы в среде разработки – Microsoft Visual Studio 2019. Для написания приложения был использован язык программирования C#. Для работы с базой данных использовался Entity Framework.

Графическая структура проекта представлена на рисунке 3.4.2.

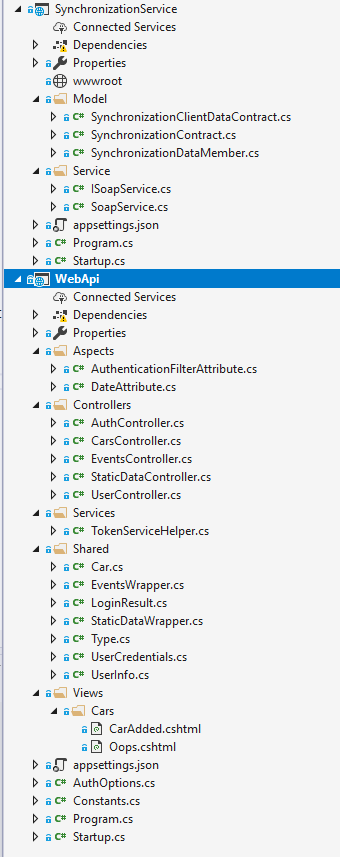
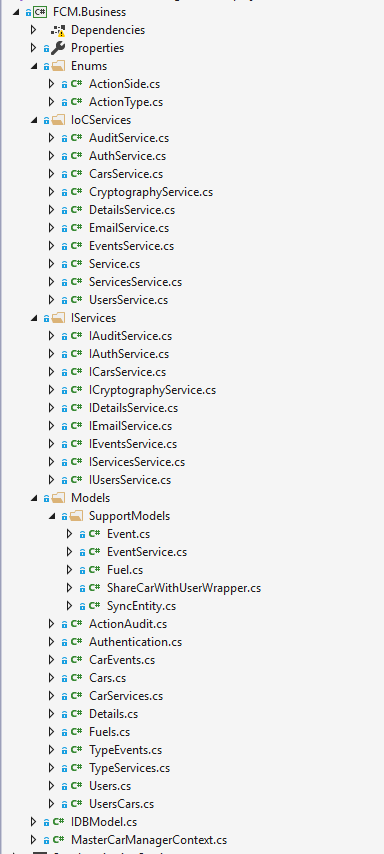


Рисунок 3.4.2 – Графическая структура проекта

Рассмотрим проект «WebApi» – предназначенный для доступа к серверу. Папка Controllers хранит в себе классы контроллеров обрабатывающие поступающие запросы к серверу, ознакомиться с методами можно в таблице 3.4.1.

**Таблица 3.4.1 – Содержание папки Controllers**

|  |  |  |
| --- | --- | --- |
| Класс AuthController – контроллер таблицы Authentification | | |
| auth/user (POST) | RegisterUser | Метод регистрации пользователя |
| auth/login (POST) | GetUserId | Метод аутентификации пользователя, возвращает токен. |
| auth/updatePassw (PUT) | UpdateUserCredentials | Метод обновления пароля |
| Класс UserController – контроллер таблицы Users | | |
| user/name/{name} (GET) | GetUserByNameAsync | Метод получения пользователя по имени |
| user/id (GET) | GetUserById | Метод получения пользователя по идентификатору |
| user (PUT) | UpdateUser | Метод обновления пользователя |
| Класс StaticDataController – контроллер обработки запросов на получение статической информации | | |
| data/getStaticInfo (GET) | GetStaticData | Метод получения статической информации |
| Класс CarsController – контроллер таблицы Cars | | |
| car/userId (GET) | GetCarsAsync | Метод получения всех машин пользователя |
| car/{carId} (GET) | GetCarById | Метод получения автомобиля по идентификатору |
| car (POST) | CreateCar | Метод добавления автомобиля |
| car/shareCar/{carId}&{email} (POST) | ShareCar | Метод предоставления другому пользователю доступа к автомобилю |
| fcm/addShareCar/{json} (GET) | AddShareCar | Метод добавления полученного автомобиля от другого пользователя в список своих автомобилей |
| fcm/deleteShareCar/{carId} (Delete) | DeleteShareCar | Метод удаления автомобиля из своего списка |
| car (PUT) | UppdateCar | Метод обновления автомобиля |
| Класс EventsController – контроллер таблиц Fuels, Services, CarEvents | | |
| events/carId/{carId:guid} (GET) | GetAllCarEventsAsync | Метод получения всех событий автомобиля |
| event/{eventId} (GET) | GetEventById | Метод получения события по идентификатору |
| fuel/{eventId} (GET) | GetFuelEventById | Метод получения события заправки по идентификатору |
| service/{eventId} (GET) | GetServiceById | Метод получения события сервиса по идентификатору |
| create/event (POST) | CreateEvent | Метод добавления события |
| create/fuel (POST) | CreateFuel | Метод добавления события заправки |
| create/service (POST) | CreateService | Метод добавления события сервиса |
| update/event (PUT) | UpdateEvent | Метод обновления события |
| update/fuel (PUT) | UpdateFuel | Метод обновления события заправки |
| update/service (PUT) | UpdateServices | Метод обновления события обслуживания |
| event/{eventId} (DELETE) | DeleteEventById | Метод удаления события |

Папка Aspects содержит атрибуты: «DateAttribute» для проверки всех дат в приложении, а так же «AuthentificationFilterAttribute» – предназначен для проверки токена, который находиться в заголовках запросов. Папка Views/Car содержит представления, для информирования пользователя о добавлении предоставленной ему машины. Папка Services содержит класс TokenServiceHepler, который предоставляет методы работы с токеном: валидация токена, получения информации из токена. Класс Constants содержит ссылку на сервер, эта ссылка в дальнейшем используется для отправки сообщений на почту пользователям, которым предоставили доступ к машине. Папка Shared содержит объекты, которые пересылаются между клиентом и сервером. Класс Program – определяет начальную точку входа в приложение. Класс Startup – определяет конфигурацию приложения.

### **3.5 Выводы по главе «Программная реализация приложений»**

В данной главе было рассмотрена разработка каждого отдельного компонента приложения.

На первоначальном этапе были рассмотрены технологии, которые применены для разработки курсового проекта.

Вторым этапом была разработка структуры базы данных на стороне сервера и клиента.

Третий этапом стала разработка клиента на мобильное устройство, где был представлен весь потенциал flutter приложения и рассмотрен алгоритм работы синхронизации со стороны клиента. Функционал для добавления фото реализовать не удалось, принятые решения по сохранению оказались не верны.

Четвертый этап – разработка серверного приложения, где была представлена архитектура приложения, структура библиотеки классов, рассмотрены Api сервера, а также был рассмотрен, алгоритм работы сервера синхронизации.

**4 Руководство пользователя**

Для того чтобы установить приложения на телефон необходимо иметь Android минимум 7.0 и версию API не ниже 26. Версию Android и API вашего телефона можно узнать в настройках телефона. Так же приложение подходит для смартфонов iOS версии не ниже 10.0.

Если версия подходит, тогда можно устанавливать приложение на мобильный телефон. В меню ваших приложений или на главном экране смартфона появится новая иконка с именем «FinanceCarManager». Запустим приложение нажатием на иконку бортового журнала автомобиля. Первым делом откроется страница запуска приложения, на которой представлено название приложения, форма для регистрации и авторизации пользователя, а также ссылка на восстановление пароля (рис. 4.1).

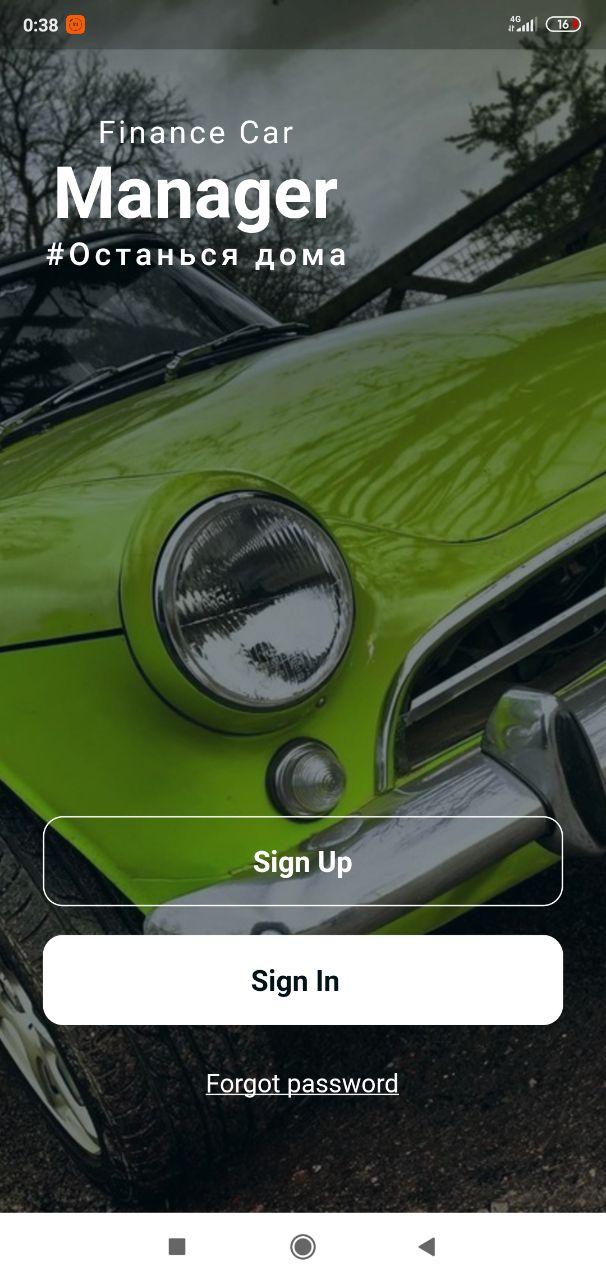


Рисунок 4.1 – страница запуска приложения

Для начала работы с приложением, новым пользователям необходимо зарегистрироваться, нажав на кнопку «Sign Up» и введя данные email-а и придумав надёжный пароль. Затем войти в мобильное приложение, нажав на кнопку «Sign In» и заполнив форму входа (рис. 4.2).

|  |  |
| --- | --- |
|  |  |

Рисунок 4.2 – страница регистрации и авторизации

После ввода корректных данных необходимо нажать на кнопку «Login», чтобы открыть главную страницу приложения. На данной странице представлены разделы меню бортового журнала автомобиля, такие как «All», «Planner», «Fuel», «Service», «Others». В верхнем левом углу расположена иконка, а также имя пользователя. В левом нижнем углу расположена кнопка для добавления новых событий, содержащих информацию о топливе, сервисе или другом типе события. Все компоненты главной страницы интерактивные, а также все кнопки являются кликабельными и содержат различные функции. Главная страница представлена на рисунке 4.3.

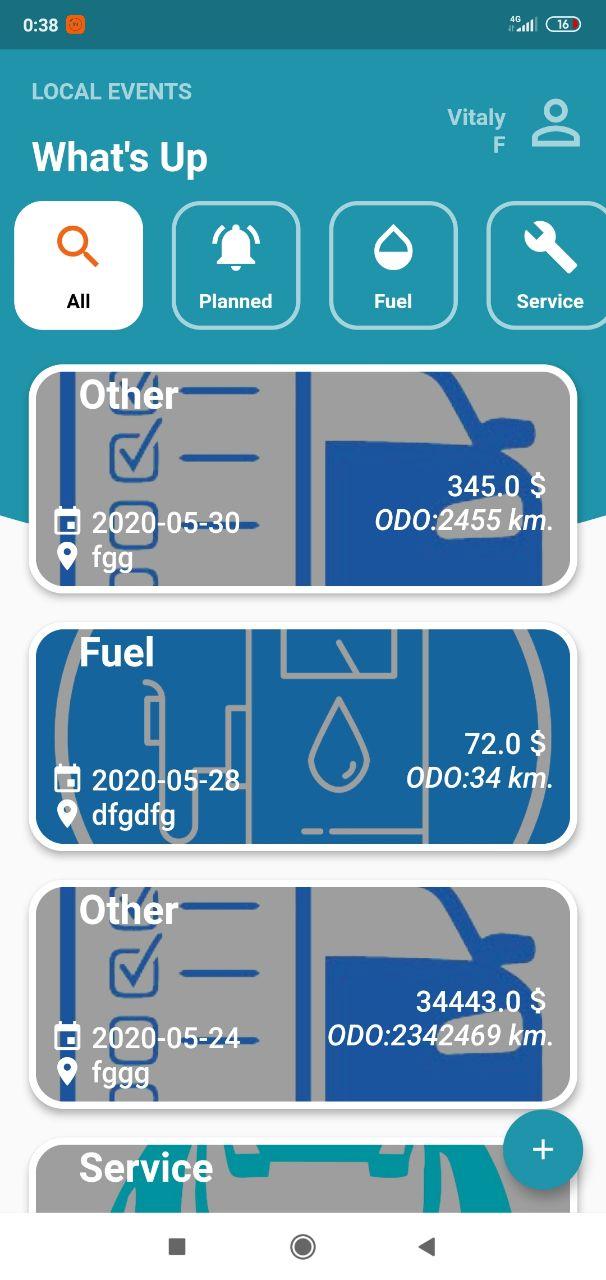


Рисунок 4.3 – главная страница приложения

При нажатии на иконку с фотографией пользователя на главной странице, открывается личный кабинет, в котором представлены его автомобили, при нажатии на кнопку в левом нижнем углу можно добавить новые транспортные средства. Также есть страница, содержащая личные данные пользователя – владельца автомобиля. Эти данные содержат такие поля, как фамилия, имя, отчество, год рождения, пол, номер телефона, город и адрес проживания. Также присутствует возможность добавления фотографии владельца формата .png, так как это компактный размер изображений и поддерживает сжатие файла без или с незначительными потерями. Изображение сохраняет исходное качество вне зависимости от интенсивности сжатия. Личный кабинет пользователя представлен на рисунке 4.4.

|  |  |
| --- | --- |
|  |  |

Рисунок 4.4 – личный кабинет пользователя

При нажатии на кнопку добавления нового транспортного средства, перед пользователем открывается страница с полями для ввода данных об автомобиле. Данные включают в себя информацию о марке и модели автомобиля, типе топлива, годе выпуска, мощности, типа коробки передач, количестве лошадиных сил и идентификационном номере машины. Также есть дополнительное поле для заметок об автомобиле и возможность добавления фотографии. После сохранения данных пользователь переходит на страницу автомобиля, где представлена сохранённая информация рисунок 4.5, а также поле для ввода email-а другого пользователя, которому можно отправить добавленное транспортное средство.

|  |  |
| --- | --- |
|  |  |

Рисунок 4.5 – добавление автомобиля

На рисунке 4.6 представлен внешний вид разделов приложения, которые содержат списки событий и функции добавления.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 4.6 – разделы приложения

При нажатии на кнопку в нижней части приложения, появляется возможность добавления нового события (рисунок 4.7).

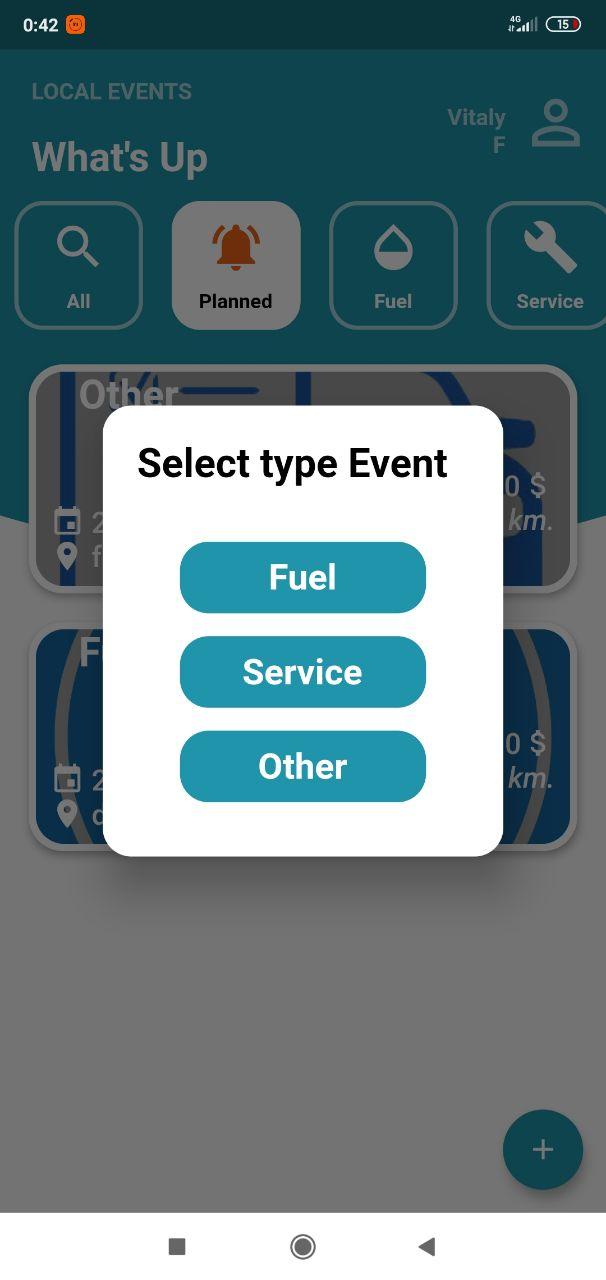


Рисунок 4.7– добавление события

После выбора типа события открывается форма для заполнения подробной информации. В зависимости от типа события формы содержат разные поля для заполнения.

Например, в форме информации о топливе, пользователю доступны такие поля, как тип топлива, стоимость, а также дата заправки автомобиля.

В форме с информацией о сервисе содержатся данные о стоимости работ, дату проведения, сведения о диагностике, а также адрес.

В форме «Другое» хранится информации о пользовательском типе события и присутствует обязательное поле «Комментарий» для названия события.

Данные формы представлены на рисунке 4.8.

|  |  |  |
| --- | --- | --- |
|  |  |  |

Рисунок 4.8– добавление данных о событиях

После добавления события и заполнения формы с информацией о нём, пользователю необходимо нажать на кнопку «Save» для сохранения данных. После этого перед ним открывается страница с информацией о текущем событии, а также с диаграммой статистики, автоматически построенной на основе введённых пользователем данных. Вид страницы приведён на рисунке 4.9.

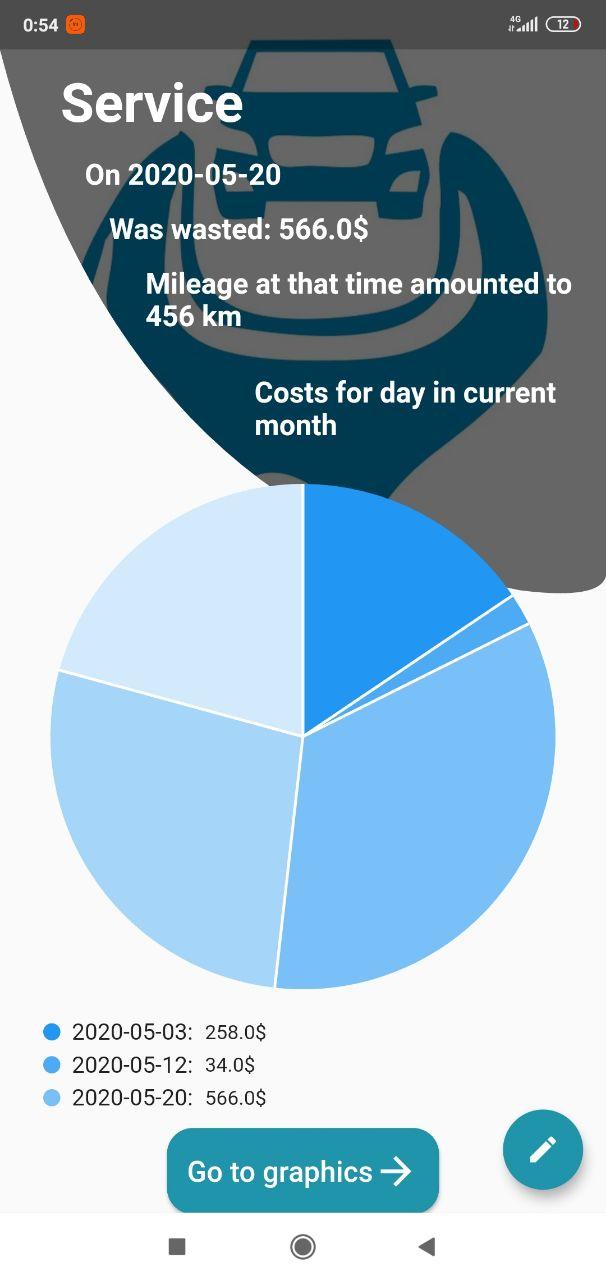


Рисунок 4.9– страница события

При нажатии на кнопку «Go to graphics» пользователь переходит в раздел с общей статистикой, представленной в виде графиков и диаграмм. Каждый график и диаграмма строится автоматически на основе введённых владельцем данных, и содержит такую информацию, как месячное количество расходов на обслуживание транспортного средства, или дневная стоимость сервиса в текущем месяце. Вид данной страницы приведён на рисунке 4.10.

|  |  |
| --- | --- |
|  |  |

Рисунок 4.10– статистика в виде графиков

На заднем фоне расположено изображение дороги, что ещё больше дополняет дизайн приложения, предназначенного для автомобилистов.

### **4.1 Выводы по главе «Руководство пользователя»**

В данной главе было описано руководство пользователя. Оно позволяет пользователю узнать требования для установки приложения на свой смартфон, а также информацию обо всех возможностях данной программы, как использовать все функции приложения. Были предоставлены скриншоты самого приложения с подробным описанием действий, для большей наглядности использования.

## **ЗАКЛЮЧЕНИЕ**

Целью моего курсового проекта является создание приложения для учёта затрат и расходов на обслуживание автомобиля, а также хранения всех данных о предстоящих событиях, сервисе, характеристиках, пробеге автомобиля в одном месте. Также были представлены графики и диаграммы затрат и пробега.

Были решены следующие поставленные задачи:

* изучение аналогов;

В результате анализа аналогов было рассмотрено 3 приложения. Были выявлены их достоинства и недостатки. Был сделан вывод о том, что очень важно не только правильно подобрать нужный функционал, но и грамотно поместить его, чтобы не перегрузить приложение. Также важна логичная и простая навигация, лёгкий и ненавязчивый дизайн.

* проектирование приложения;

В бортовом журнале автомобиля был определен основной функционал приложения. При проектировании базы данных были определены необходимые таблицы и разработана схема базы данных. В серверном приложении был определен архитектурный стиль взаимодействия между клиентом и сервером (REST), методы для работы с серверной частью. Также был спроектирован сервер синхронизации, основанный на формате протокола SOAP.

* реализация приложения;

Данная задача была реализована с помощью следующих технических средств: MS SQL, SqfLite, сред разработки: Visual Studio Code и Microsoft Visual Studio, языков программирования Dart и C#, модульной платформы .NET Core и мобильной платформы Flutter, библиотек Entity Framework, SOAP, JwtBearer.

При решении данной задачи была достигнута поставленная цель, и был разработан проект «Бортовой журнал автомобиля», который включает в себя мобильное приложение, сервер и удалённую базу данных.

* разработка руководства пользователя.

Разработано руководство, предоставляющее пользователям ознакомление со всеми возможностями программы, а также инструкцию по использованию функций данного приложения.

Мобильное приложение имеет простой, интуитивно понятный пользователю интерфейс и обладает следующими особенностями:

* добавление и редактирование событий, связанных с автомобилем;
* просмотр всех событий по категориям;
* добавление автомобилей;
* предоставления доступа к автомобилю другому пользователю;
* синхронизация данных между несколькими пользователями приложения;
* формирование отчётов о расходах в виде графиков и диаграмм;

В соответствии с поставленными задачами и полученным результатом можно сделать вывод, что курсовой проект выполнен в полном объеме и поставленные задачи успешно выполнены.

## **Список использованных источников**

1 Исследовательская работа «Автомобиль: вчера, сегодня, завтра» [Электронный ресурс]. / infourok.ru – Режим доступа: https://infourok.ru/issledovatelskaya-rabota-avtomobil-vchera-segodnya-zavtra-1918923.html – Дата доступа: 13.09.2020.

1. Flutter. Основы [Электронный ресурс] / habr.com/ru. – Режим доступа: https://habr.com/ru/post/430918/. Дата доступа: 27.09.2020
2. Используем SQLite в Flutter [Электронный ресурс] / habr.com/ru. – Режим доступа: https://habr.com/ru/post/435418/. Дата доступа: 13.10.2020
3. Архитектура REST [Электронный ресурс] / habr.com/ru. – Режим доступа: https://habr.com/ru/post/38730/. Дата доступа: 15.11.2020
4. Введение в SOAP [Электронный ресурс] – quizful.net – Режим доступа: http://www.quizful.net/post/soap-simple-object-access-protocol – Дата доступа: 15.11.2020