Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет Математики и информатики
Кафедра Информационных и управляющих систем
Направление подготовки 09.03.01 - «Информатика и вычислительная техника»
Направленность (профиль) образовательной программы Автоматизированные системы обработки информации и управления

ДОПУСТИТЬ К ЗАЩИТЕ
Зав. Кафедрой
_____ А.В. Бушманов
«____ » _____ 2023г

БАКАЛАВРСКАЯ РАБОТА

на тему: Разработка мобильного приложения для сервисного обслуживания автомобиля

Исполнитель студент группы 953об	_____ (подпись, дата)	Н.В. Данилов
Руководитель доцент, канд. техн. наук	_____ (подпись, дата)	Л.В. Никифорова
Консультант по безопасности и экологичности доцент, канд. техн. наук	_____ (подпись, дата)	А.Б. Булгаков
Нормоконтроль инженер кафедры	_____ (подпись, дата)	В.Н. Адаменко

Благовещенск 2023

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ФГБОУ ВО «АмГУ»)

Факультет Математики и информатики

Кафедра Информационных и управляющих систем

УТВЕРЖДАЮ
Зав. Кафедрой
_____ А.В. Бушманов
«____ » _____ 2023г

ЗАДАНИЕ

К выпускной квалификационной работе студента: Н.В. Данилов

1. Тема выпускной квалификационной работы: Разработка мобильного приложения для сервисного обслуживания автомобиля

(утверждена приказом от _____ № _____)

2. Срок сдачи студентом законченной работы (проекта):

3. Содержание выпускной квалификационной работы: анализ предметной области; освоение программного и технического обеспечения; разработка алгоритм решения; применение результата на практике.

4. Перечень материалов приложения: физическая структура базы данных.

5. Дата выдачи задания:

Руководитель выпускной квалификационной работы: Никифорова Л.В. доцент кафедры ИБ, канд. техн. Наук.

(фамилия, имя, отчество, должность, уч. степень, уч. звание)

Задание принял к исполнению (. .):

(Подпись студента)

РЕФЕРАТ

Дипломная (бакалаврская) работа содержит 80 страниц, 26 рисунков, 23 таблицы, 1 приложение, 11 источников.

.NET MAUI, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, ТЕХНИЧЕСКОЕ ОБСЛУЖИВАНИЕ АВТОМОБИЛЯ, SQLITE, OBJECT-RELATIONAL MAPPING.

Цель разработки мобильного приложения для сервисного обслуживания автомобилей заключается в создании удобной и эффективной платформы, которая поможет пользователям получить качественное и своевременное обслуживание для своих автомобилей. Для достижения этой цели необходимо провести исследование и анализ процессов и требований, связанных с сервисным обслуживанием автомобилей.

В ходе работы требуется разработать функциональность, которая позволит пользователям заказывать сервисные услуги, записываться на техническое обслуживание. Приложение должно обеспечивать удобный интерфейс для просмотра и выбора доступных услуг, а также возможность оплаты через приложение.

Результатом данной работы является мобильное приложение для сервисного обслуживания автомобилей, которое предоставляет пользователям простой и надежный способ получить необходимые услуги. Приложение позволяет пользователям управлять процессом обслуживания своего автомобиля, экономя время и упрощая взаимодействие с автосервисом. учитывая требования и потребности клиентов.

СОДЕРЖАНИЕ

НОРМАТИВНЫЕ ССЫЛКИ

В настоящей бакалаврской работе использованы ссылки на следующие стандарты и нормативные документы:

СП 2.2.3670-20. Санитарно-эпидемиологические требования к условиям труда.

СанПиН 1.2.3685-21. Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания.

ГОСТ Р 50948-2001. Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности.

ГОСТ Р 50949-2001. Средства отображения информации индивидуального пользования. Методы измерений и оценки эргономических параметров и параметров безопасности.

ГОСТ 28406-89. Персональные электронные вычислительные машины. Интерфейсы видеомониторов. Общие требования

ГОСТ Р 51645-2017. Рабочее место для инвалида по зрению типовое специальное компьютерное. Технические требования к оборудованию и производственной среде.

ГОСТ Р ИСО 1503-2014. Эргономика. Требования к пространственной ориентации и направлениям движения органов управления.

ГОСТ 12.1.007-76 Система стандартов безопасности труда (ССБТ). Вредные вещества. Классификация и общие требования безопасности.

ГОСТ Р 70146-2022 Национальный стандарт Российской Федерации. Ресурсосбережение. Отходы электроники и электробытовой техники.

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

ТЗ – Техническое задание

БД – База данных

ИС – Информационная система

АСОИУ – Автоматизированная система обработки информации и управления

СУБД – Система управления базами данных

API – Application Programming Interface

MAUI – Multi-Platform App UI

API (Application Programming Interface) – это набор программных инструкций и протоколов, предоставляемых разработчиками, для взаимодействия и интеграции между различными программами, сервисами или компонентами приложений. API определяет способы, как одна программа может взаимодействовать с другой, определяя доступные функции, данные и способы коммуникации между ними.

Object-Relational Mapping (ORM) – это технология программирования, предоставляющая абстракцию и средства взаимодействия между объектно-ориентированными приложениями и реляционными базами данных. ORM обеспечивает автоматическое отображение объектов приложения на соответствующие таблицы в базе данных, а также управление операциями чтения, записи и изменения данных.

MAUI (Multi-platform App UI) – это фреймворк для разработки кроссплатформенных мобильных приложений, который предоставляет разработчикам средства для создания пользовательского интерфейса, работающего на различных операционных системах, таких как Android, iOS и Windows. MAUI позволяет разработчикам использовать общий код и ресурсы для создания приложений, которые могут работать на разных платформах, что снижает затраты на разработку и поддержку.

ВВЕДЕНИЕ

В современном мире автомобиль является неотъемлемой частью нашей жизни. Однако, чтобы автомобиль всегда находился в отличном техническом состоянии и обеспечивал безопасность и комфортную поездку, регулярное и качественное техническое обслуживание является необходимостью. С течением времени, автомобиль требует замены масла, проверки и регулировки систем, замены запчастей и выполнения других работ, чтобы оставаться надежным и безопасным средством передвижения.

В связи с этим, разработка мобильного приложения для сервисного обслуживания автомобиля становится актуальной задачей. Такое приложение поможет автовладельцам упростить процесс учета и планирования технического обслуживания, а также обеспечит контроль над финансами, связанными с обслуживанием автомобиля. Благодаря использованию мобильных устройств, пользователи смогут легко отслеживать все проведенные работы, записывать расходы на запчасти и топливо, получать уведомления о предстоящих сроках обслуживания и многое другое.

Такое мобильное приложение не только сэкономит время и усилия автовладельцев, но и поможет им поддерживать свой автомобиль в отличном состоянии. Оно предоставит возможность создания истории обслуживания, которая будет полезна при продаже автомобиля или для личного контроля. Кроме того, приложение может предоставить дополнительные функции, такие как подбор запчастей, контакты сервисных центров, советы по техническому обслуживанию и многое другое.

В контексте данной выпускной квалификационной работы, целью является разработка мобильного приложения, предназначенного для обеспечения сервисного обслуживания автомобилей. Оно будет способствовать удовлетворению потребностей автовладельцев, предоставляя им удобный и функциональный инструмент для учета и планирования технического обслуживания

своих автомобилей.

Основное назначение разработанного приложения будет заключаться в облегчении процесса учета информации о техническом состоянии автомобиля, планировании регулярных технических осмотров и обслуживания, а также в предоставлении полезных рекомендаций и напоминаний, связанных с техническим обслуживанием.

Особое внимание будет уделено созданию надежного и интуитивно понятного приложения, которое будет не только полезным, но и удобным в использовании для широкого круга автовладельцев, стремящихся поддерживать свои автомобили в отличном состоянии. Приложение будет обладать надежным функционалом и простым в использовании интерфейсом, чтобы обеспечить максимальную удобство и удовлетворение пользовательских потребностей.

Таким образом, данный проект выпускной квалификационной работы будет нацелен на создание надежного и интуитивно понятного мобильного приложения, предоставляющего удобный и функциональный инструмент для учета и планирования технического обслуживания автомобилей, пригодный для использования всеми автовладельцами, стремящимися поддерживать свои автомобили в отличном состоянии.

1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 История развития мобильных приложений

В 1998 году был представлен Wireless Application Protocol (WAP), который стал ключевым инструментом для объединения интернета и мобильной связи. Благодаря WAP-технологии стало возможным встраивать в мобильные телефоны браузеры, устанавливать соединение с серверами и получать данные на мобильные устройства.

Одним из ранних примеров телефона с WAP-браузером является Nokia 7100, выпущенный в 1999 году. Это время также стало зарождением компаний, специализирующихся на разработке продуктов специально для мобильных устройств.

WAP предоставил пользователям не только возможность играть, но и доступ к чтению новостей, использованию электронной почты, загрузке карт и даже бронированию билетов с помощью мобильных устройств. Для этого создавались специальные WAP-сайты с адаптированной разметкой для мобильных экранов, представляющие собой простые страницы с текстом и ссылками, в основном без изображений.

В 2001 году Symbian стала открытой операционной системой, и в это же время был выпущен Nokia 7650, на котором можно было устанавливать приложения от сторонних разработчиков. Это предвещало прорыв на рынке, однако развитие было затруднено сложностями разработки и ограниченными возможностями смартфонов того времени.

Разработчики столкнулись с ограниченным выбором инструментов разработки для Symbian. Основным языком программирования стал C++, который был сложным для изучения и компиляции. Создание приложений, совместимых с большинством устройств на базе Symbian, требовало дополнительных усилий. Также многих отпугивала необходимость приобретения сертификатов безопасности для подписи приложений.

В 2001 году операционная система Symbian стала открытой, и в это же время был представлен Nokia 7650, который поддерживал установку приложений от сторонних разработчиков. Ожидалось, что это станет прорывом на рынке, однако прогресс был затруднен сложностями разработки и ограниченными возможностями смартфонов.

У разработчиков был ограниченный выбор инструментов для разработки приложений под Symbian. Основным языком программирования был C++, который требовал сложного изучения и компиляции. Разработчикам приходилось преодолевать трудности, чтобы создать приложение, совместимое с большинством устройств на базе Symbian. Кроме того, многих отпугивала необходимость приобретения сертификатов безопасности для подписи приложений.

В 2007 году Стив Джобс представил первый iPhone, вызывая много обсуждений о его возможностях, но не раскрывая подробности о том, как эти возможности были достигнуты.

Архитектура операционной системы iOS была подобна MacOS, но система была полностью закрытой. Джобс не желал, чтобы сторонние разработчики создавали приложения для iOS, и не планировал открывать набор инструментов разработчика (SDK). Вместо этого он поощрял разработку веб-приложений и предоставил возможность создавать браузерные закладки на домашнем экране. Эту информацию мы узнаем из биографии Джобса, написанной Уолтером Айзексоном.

Однако, некоторые исследователи и энтузиасты смогли проникнуть в файловую систему iOS, начали разрабатывать инсталляторы для нативных приложений и, в конечном итоге, сами приложения. Так возникло явление джейлбрейка.

Позже, совет директоров компании Apple все же убедил Джобса в необходимости легализации сторонних приложений. В результате, в марте 2008 года стал доступен iPhone SDK для всех желающих, а в июле был представлен

App Store. Это означало, что Apple взяла на себя роль дистрибьютора разработчиков для пользователей.

App Store стал катализатором развития приложений, но одной из проблем был язык программирования Objective-C. Этот язык значительно отличался от популярных скриптовых языков JavaScript и Flash Action Script, которые были широко распространены на тот момент. Мало кто желал тратить время на изучение нового синтаксиса, особенно учитывая, что устройства на iOS занимали небольшую долю рынка, а основная часть рынка принадлежала смартфонам на базе Symbian.

В 2008 году появился Android Market, а в 2010 году появился Windows Mobile Store. К тому времени мобильный интернет стал более доступным, а инструментарий разработки (SDK) для разных платформ начал предлагать решения для безопасности и интеграции. Это знаменовало новую фазу развития в области разработки приложений: от развлекательных проектов разработчики перешли к решению бизнес-задач. Однако, на горизонте уже маячила другая проблема.

Быстрый параллельный рост iOS и Android привел к появлению двухполярной системы, и разработчикам приходилось поддерживать несколько платформ одновременно. На тот момент были доступны только ограниченные кроссплатформенные инструменты, такие как Flash и стандартные мобильные браузеры. Тем не менее, в 2010 году Apple отказалась поддерживать технологию Adobe Flash в iOS.

Разработка даже самых простых приложений под разные платформы требовала значительных человеческих, временных и финансовых ресурсов. Хотя браузерные технологии были достаточно хорошо развиты, мобильная веб-разработка замедлялась из-за низкой производительности смартфонов.

На помощь пришли библиотеки компонентов и фреймворки для создания приложений на Android и iOS, использующие браузерные технологии без необходимости использования языков программирования. Среди таких

инструментов можно назвать Xamarin, Cordova и Phonegap.

С использованием указанных инструментов создается аналог мобильного веб-сайта, к которому применяется платформенный код для обеспечения взаимодействия между системой и приложением.

Эти инструменты решают проблемы небольших приложений с ограниченным функционалом и предоставляют возможность бизнесу быстро и с небольшими затратами проверить свое присутствие на мобильных платформах. Однако далее необходимо обратить внимание на нативную разработку, поскольку возникают проблемы с производительностью, потреблением ресурсов, отзывчивостью кроссплатформенных приложений и несоответствием их дизайна стандартам платформы.

1.2 История развития автомобильной электроники

Параллельно с развитием механических систем автомобилей, инженеры постоянно стремились интегрировать электронные компоненты, делая автомобили более безопасными, управляемыми и интеллектуальными. Сегодняшняя ситуация предоставляет все необходимые предпосылки для этого: IT-индустрия развивается с огромной скоростью, автопроизводители готовы к сотрудничеству и ведут перспективные исследования, а корпорации вкладывают средства в развитие автотранспорта. Однако, эволюция «умных» автомобилей происходила постепенно на протяжении более чем полувека, принимая различные формы и концепции, от безопасности до развлечений.

Первым технологическим прорывом в автомобилестроении стало появление интереса автомобильных компаний к электрическим стартерам, которые впервые были установлены в 1911 году. Затем нововведения начали касаться удобства водителя и даже его развлечений за рулем: в 1925 году был представлен прикуриватель, в 1930 году – радио, в 1956 году – усилитель руля, в 1970 году – кассетный плеер, в 1984 году – системы пассивной безопасности в виде надувных подушек. В следующем году появились CD-проигрыватели, в 1994 году – компьютерная диагностика автомобиля на приборной панели, в 1995

году – системы глобального позиционирования (GPS), в 2000 году – порты USB и Bluetooth, первые шаги к «подключенному» автомобилю.

В середине XX века появился первый опыт создания «умного» автомобиля. General Motors Firebird II, выпущенный в 1956 году, был четырехместным автомобилем с независимой подвеской. Под своим титановым корпусом он имел газотурбинный двигатель Whirlfire GT-304 мощностью 200 л.с., электропакет и интегрированную систему кондиционирования воздуха, сравнимую с технологиями начала XXI века. Firebird II продолжил дизайн и эргономику предыдущей версии автомобиля 1953 года, который был назван "реактивным самолетом на колесах" и вдохновлен концепциями истребителей того времени. Однако Firebird II впервые воплотил в себе концепцию будущих шоссе - сложную систему управления, взаимодействующую с электрическим проводом, встроенным в дорогу, чтобы передавать сигналы и служить ориентиром для передовых автомобилей. Предполагалось, что электромагнитное поле поможет минимизировать опасные ситуации на дороге, сократив роль человека в управлении. Эта смелая модель произвела фурор на выставках, но так и не вышла в серийное производство.

В то время в Европе и США строились шоссе будущего. Первым серийным автомобилем, который реально взаимодействовал с такими дорогами, стал легендарный Citroen DS. Он занял третье место в рейтинге автомобилей века. Хотя его 75-сильный двигатель не был особенно мощным для того времени, автомобиль отличался передовой трансмиссией, объединенной с рулевым управлением, тормозами и гидропневматической подвеской. Эта конструкция опередила свое время на много лет вперед. Citroen DS мог взаимодействовать с дорогой с помощью электрического сигнала, но не было речи о самостоятельном автопилоте – это было больше развлечением. Однако его невероятная популярность, передовые технологии и, хоть и иллюзорный, автопилот, сделали этот Citroen похожим на летающий автомобиль Фантомаса.

В 60-70 годах проводились эксперименты с бортовыми компьютерами,

но они так и не вошли в серийное производство. Один из таких экспериментальных автомобилей был Chrysler Plymouth, который был оснащен относительно примитивным бортовым компьютером, занимающим половину заднего сиденья, и генератором, установленным на крыше автомобиля для питания системы. Лабораторные испытания этого автомобиля проводились в течение 10 лет, но не возникло никаких планов на его серийное производство.

Однако как инженеры, так и фантазеры непрерывно стремились создать автомобили, которые были бы не только средством передвижения и роскошью, но и умными помощниками, способными облегчить жизнь и сделать дороги безопасными. Это стремление нашло отражение в кино, где после нескольких фильмов с «говорящими» автомобилями настоящими хитами стали серия фильмов о Джеймсе Бонде с его передовыми автомобилями и, конечно, легендарный "Рыцарь дорог". Автомобиль КИТТ на базе Pontiac Firebird Trans AM был умным, обладал чувством юмора и не только развивал скорость до 500 км/ч, был почти неуязвимым, но и умел разговаривать, ездить на полном автопилоте и контролировать все электронные устройства на расстоянии.

Организация AUTOSAR (AUTomotive Open System ARchitecture) работает над созданием стандартизированной открытой структуры программного обеспечения для автомобильной электроники, за исключением информационно-развлекательных систем. Это программное обеспечение должно быть масштабируемым, локализируемым, соответствовать требованиям безопасности и быть поддерживаемым на протяжении всего срока службы автомобиля. Стандарт AUTOSAR распространяется на электронику кузова, силовой агрегат, шасси, системы безопасности, мультимедийные системы, телематику и интерфейс взаимодействия водителя с автомобилем.

Протокол бортовой электроники FlexRay представляет собой высокоскоростной сетевой протокол для автомобилей, разработанный консорциумом FlexRay, включающим компании NXP, BMW, DaimlerChrysler, Bosch, GM и Volkswagen. Скорость передачи данных по этому протоколу достигает 10

Мбит/с, что в десятки раз быстрее, чем современная шина CAN и устаревший диагностический протокол OBD. Контроллеры FlexRay используются для контроля состояния различных частей автомобиля, где надежность диагностики критически важна, таких как двигатель, трансмиссия, подвеска, тормозная система и рулевое управление. Протокол также предназначен для расширения возможностей бортового управления.

Стандарт Automotive Safety Restraints Bus (ASRB 2.0) определяет спецификацию для электронных систем автомобиля, отвечающих за физическую безопасность водителя и пассажиров.

Автомобильные системы автопилотирования, автопарковки и навигации являются программным и аппаратным обеспечением, которое становится неотъемлемой частью вождения. Более того, эти системы уже сейчас выполняют функции безопасности и защиты, например, вызов спецслужб в случае серьезного ДТП, а в будущем их функциональность будет только усиливаться.

Применение типичных решений интернета вещей (IoT) также находит свое применение в автомобилях. Например, GM сотрудничает с IBM для использования системы Watson в умных автомобилях.

Одной из главных проблем программного обеспечения для автомобилей является необходимость учета аппаратных особенностей, которые могут использоваться более десяти лет. Поэтому важно иметь передовые возможности обновлений ПО, а лучше всего иметь программное обеспечение, опережающее свое время.

1.3 Актуальность использования

Разработка мобильного приложения для отслеживания технического обслуживания автомобиля является важной и необходимой для многих автовладельцев. Вот несколько основных причин, почему такое приложение имеет высокую ценность.

Во-первых, регулярное техническое обслуживание является неотъемлемой частью заботы о транспортном средстве. Автомобиль состоит из

множества сложных механических и электрических компонентов, которые требуют регулярного обслуживания, чтобы они работали эффективно и безопасно. Приложение позволяет автовладельцам следить за состоянием и сроками обслуживания различных компонентов автомобиля, таких как двигатель, тормозная система, подвеска, фильтры и многое другое.

Во-вторых, мобильное приложение обеспечивает удобный способ планирования и организации технического обслуживания. Пользователи могут создавать расписания обслуживания, записывать даты и мили, когда необходимо выполнить определенные работы. Приложение может предупреждать автовладельцев о предстоящих или просроченных обслуживаниях, чтобы они не пропустили важные процедуры и не столкнулись с проблемами в будущем.

Третья причина связана с сохранением истории обслуживания автомобиля. Приложение позволяет автовладельцам вести подробные записи о всех выполненных работах, замененных запчастях, проведенных диагностических проверках и других процедурах. Это может быть полезно в случае продажи автомобиля или при необходимости предоставить информацию механику или страховой компании. Кроме того, ведение истории обслуживания помогает в поддержании регулярности и своевременности технического обслуживания.

Четвертая причина касается экономии времени и удобства. Мобильное приложение предоставляет удобный и легкий способ доступа к информации об автомобиле и его обслуживании. Пользователи могут легко проверять расписание обслуживания, получать уведомления и предупреждения о необходимости замены или ремонта, а также осуществлять контроль расходов на обслуживание автомобиля.

В целом, мобильное приложение для отслеживания технического обслуживания автомобиля обеспечивает автовладельцам лучший контроль над состоянием своего транспортного средства, помогает сохранить безопасность и производительность автомобиля, предоставляет удобство в организации и планировании обслуживания, а также позволяет вести детальную историю

работ и расходов.

1.4 Обзор существующих методов решения рассматриваемой задачи

1.4.1 Car Xpenses light

Car Xpenses light представляет собой мобильное приложение для обслуживания автомобиля на платформах Android и IOS. Оно предоставляет автовладельцам возможность эффективно вести учет и контролировать все аспекты технического обслуживания своего автомобиля, а также отслеживать расходы на обслуживание автомобиля. Сайт приложения представлен на рисунке 1.

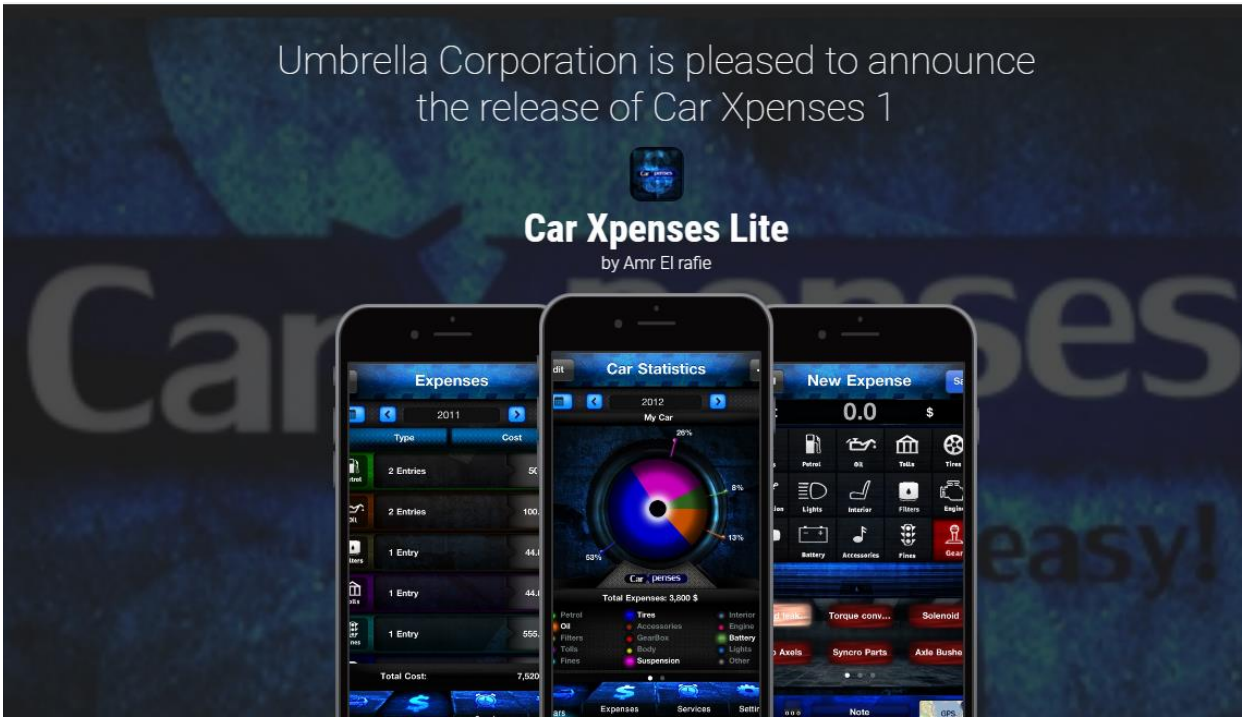

Рисунок 1 – Сайт Car Xpenses Lite

Одной из основных функций приложения Car Xpenses Light является возможность записи и отслеживания информации о проведенных работах на автомобиле. Пользователи могут создавать записи о регулярном обслуживании, замене масла, технических ремонтах и других видах обслуживания. Каждая запись может содержать дату, описание работ, стоимость, пройденный пробег и другие полезные детали. Таким образом, автовладельцы имеют полный контроль над историей технического обслуживания своего автомобиля.

Помимо этого, Car Xpenses Light предоставляет возможность отслеживания расходов на обслуживание автомобиля. Пользователи могут добавлять записи о затратах на запчасти, топливо, страховку, автомойку и другие траты, связанные с автомобилем. Приложение автоматически суммирует расходы и предоставляет пользователю общую статистику о затратах за определенный период времени. Это позволяет автовладельцам контролировать свои финансы и планировать бюджет на обслуживание автомобиля.

Car Xpenses Light также обладает полезными функциями напоминаний и уведомлений. Пользователи могут установить напоминания о предстоящих регулярных обслуживаниях, смене масла или замене деталей. Это помогает избежать упущения сроков и обеспечивает своевременное проведение необходимых работ на автомобиле.

В качестве положительных сторон можно выделить:

– удобство использования;

– отслеживание истории обслуживания;

– управление расходами;

– напоминания и уведомления.

Из негативных сторон Car Xpenses Light можно выделить:

– ограниченный функционал;

– отсутствие синхронизации данных;

– ограниченная поддержка.

1.4.2 Приложение «Мой авто»

Мой авто – это удобное и полезное мобильное приложение, которое предназначено для учета и обслуживания автомобиля. Это приложение обладает множеством функций, которые позволяют пользователям эффективно управлять своим автомобилем и поддерживать его в отличном техническом состоянии.

Одной из ключевых возможностей приложения Мой авто является ведение детального учета всех операций и событий, связанных с автомобилем.

Пользователи могут записывать информацию о заправках, пробеге, заменах масла, плановых и внеплановых ремонтах, а также других видов обслуживания. Это позволяет автовладельцам иметь полный контроль над историей эксплуатации своего автомобиля и более точно планировать дальнейшие операции. Сайт приложения представлен на рисунке 2.

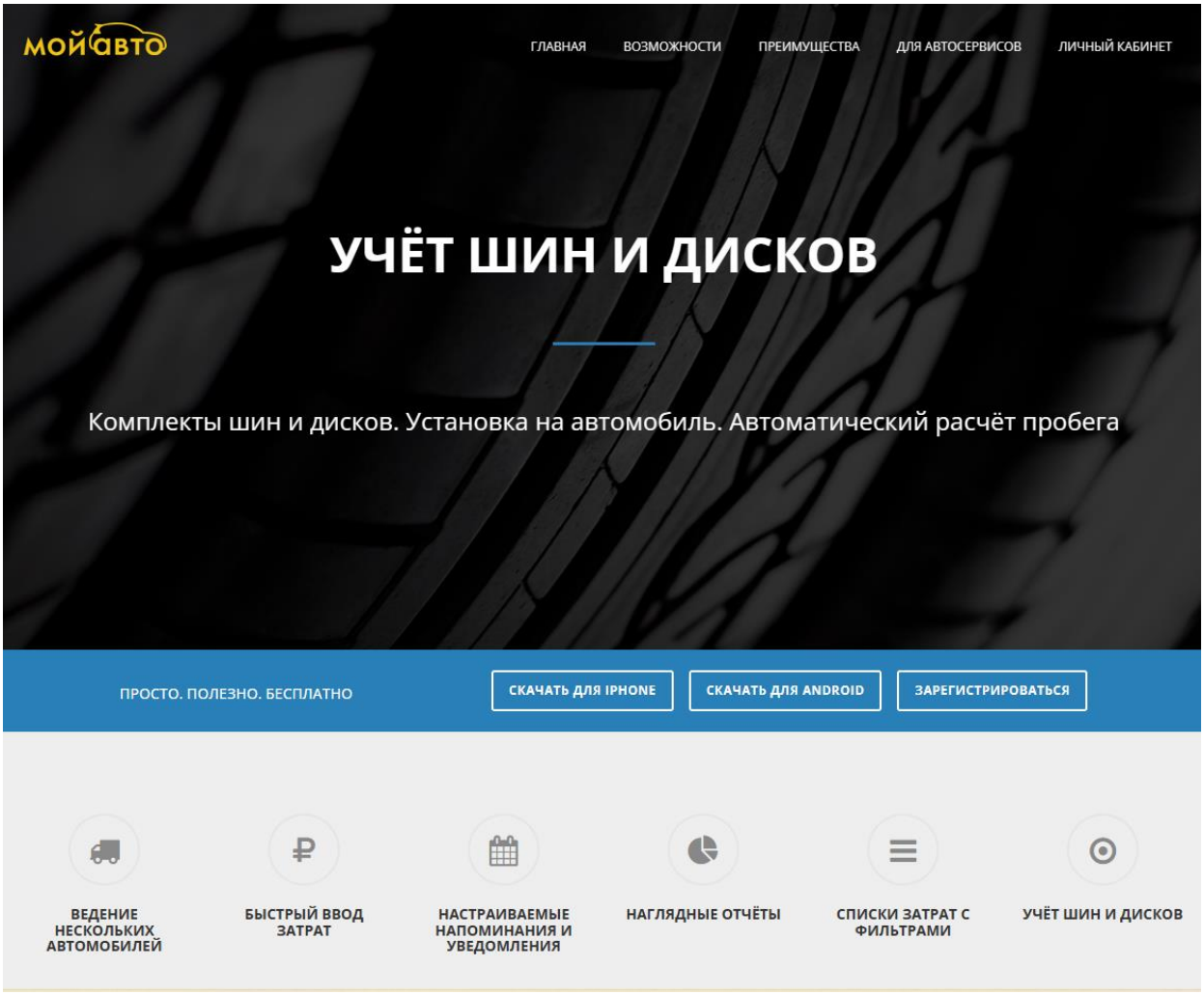

Рисунок 2 – Сайт приложения «Мой авто»

Мой авто также предоставляет возможность отслеживания расходов на обслуживание автомобиля. Пользователи могут вводить информацию о затратах на топливо, запчасти, ремонтные работы, страховку и другие расходы, связанные с автомобилем. Приложение автоматически суммирует эти расходы и предоставляет пользователю общую статистику, что помогает контролировать финансовые затраты и позволяет лучше планировать бюджет.

Дополнительно, Мой авто предлагает различные напоминания и уведомления, которые помогают автовладельцам не пропустить важные события и сроки. Приложение может отправлять уведомления о необходимости смены масла, прохождении технического осмотра, плановом обслуживании и других важных моментах, что помогает поддерживать автомобиль в оптимальном состоянии.

Интерфейс Мой авто является интуитивно понятным и удобным для использования. Он предлагает пользователю простую навигацию и легкую доступность к функциям приложения (рис. 3). Пользователи могут быстро и легко добавлять новые записи, просматривать статистику и управлять своим автомобилем через интуитивно понятные элементы управления.

Рисунок 3 – Функциональные возможности приложения

Одним из преимуществ Мой авто является его возможность синхронизации данных через облачное хранилище. Это позволяет пользователям сохранять свою информацию на сервере и иметь доступ к ней с разных устройств. Синхронизация также обеспечивает безопасное хранение данных и защиту от потери информации в случае сбоя устройства.

Однако, следует отметить, что приложение Мой авто имеет свои ограничения. Некоторые пользователи считают его функционал недостаточным,

особенно если требуется более сложное управление автомобилем или анализ статистики. Также возможно, что некоторые функции могут быть доступны только в платной версии приложения.

В целом, «Мой авто» – это полезное и удобное приложение для ведения учета и обслуживания автомобиля. Оно помогает пользователям поддерживать свой автомобиль в отличном состоянии, планировать финансы и быть в курсе всех необходимых операций и событий, связанных с автомобилем.

1.5 Формулировка задачи исследования и общей методики ее решения

Целью данного исследования является разработка мобильного приложения для сервисного обслуживания автомобиля, которое будет обеспечивать удобный и эффективный способ учета и планирования технического обслуживания автомобиля, а также предоставлять дополнительные функциональные возможности, улучшающие опыт пользователей.

Общая методика решения задачи:

– провести исследование и определить основные требования и потребности пользователей в отношении приложения для сервисного обслуживания автомобиля. Учесть функциональные и пользовательские требования, включая учет технических данных, планирование обслуживания, финансовый учет и другие возможности;

– разработать архитектуру и дизайн мобильного приложения, определить функциональные модули, взаимодействие с базой данных и сторонними сервисами, а также создать пользовательский интерфейс, который будет интуитивно понятным и удобным в использовании;

– на основе проектного решения, разработать мобильное приложение для сервисного обслуживания автомобиля, используя подходящий фреймворк или инструментарий, такой как .NET MAUI или Xamarin. Реализовать функциональные модули, обеспечить интеграцию с базой данных, реализовать логику бизнес-процессов и обработку пользовательских действий;

– провести тщательное тестирование приложения для выявления и устранения возможных ошибок, проверить его работоспособность, производительность и безопасность. Выполнить отладку и корректировку кода при необходимости;

– подготовить мобильное приложение для развертывания на целевых устройствах. Обеспечить совместимость с различными операционными системами (Android, iOS) и устройствами разных разрешений экрана. Развернуть приложение на мобильных платформах для широкого доступа пользователей;

– провести оценку и анализ эффективности приложения на основе обратной связи пользователей и собранных метрик использования. Выявить проблемы и потенциальные улучшения, чтобы продолжить развитие и совершенствование приложения.

Таким образом, данная исследовательская работа будет ориентирована на разработку и реализацию мобильного приложения для сервисного обслуживания автомобиля с использованием определенной методики, которая обеспечит учет технического обслуживания, планирование работ, управление расходами и другие функциональные возможности для оптимизации процесса обслуживания автомобилей.

2 ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

Проблема ведения учета технического обслуживания автомобиля заключается в неэффективной организации и отслеживании информации о проведенных работах, замене деталей и расходах на обслуживание. Это может привести к ряду негативных последствий.

Во-первых, без достаточно структурированного учета может возникнуть запутанность и неясность данных. Отсутствие четкой системы организации информации приводит к трудночитаемому и неудобному представлению данных о проведенных работах, что затрудняет анализ и принятие информированных решений.

Во-вторых, отсутствие систематического учета может привести к потере важных данных. Без централизованной системы, которая бы хранила и отслеживала информацию о проведенных работах и замененных деталях, может быть сложно вспомнить историю обслуживания автомобиля, особенно при необходимости предоставления этой информации механику или страховой компании.

Кроме того, отсутствие системы уведомлений и напоминаний о предстоящем обслуживании может привести к упущению сроков и задержкам в проведении необходимых работ. Это может повлиять на надежность и безопасность автомобиля, а также привести к увеличенным расходам на ремонт и замену деталей.

Наконец, без систематического учета и анализа данных о проведенном обслуживании автовладелец может упустить возможности для оптимизации расходов и повышения эффективности обслуживания. Недостаток информации о расходах и выполненных работах затрудняет анализ трендов, выявление потенциальных проблем и принятие информированных решений для улучшения обслуживания автомобиля.

Все эти проблемы подчеркивают важность разработки системы ведения

учета технического обслуживания автомобиля, которая бы обеспечивала структурированное хранение информации, систематические уведомления и возможность анализа данных.

2.1 Средства разработки программного обеспечения

Для определения наиболее подходящего языка программирования в контексте разработки мобильного приложения из рассмотренных альтернатив, а именно Java, C++, Python, PHP, Go и C#, проведен анализ по нескольким критериям, указанных в таблице 1. Каждый критерий был оценен по шкале от 1 до 5, где 1 представляет низкий уровень, а 5 – высокий уровень.

Критерии, учтенные в анализе, включают синтаксис и читаемость, производительность, экосистему и сообщество, масштабируемость, удобство разработки, безопасность и удобство для веб-разработки.

После проведения анализа, были подсчитаны суммы оценок для каждого языка, что позволяет сравнить их и определить предпочтительный язык программирования.

Таблица 1 – Анализ языков программирования

Критерий	**Python**	**Java**	**C++**	**PHP**	**GO**	**C#**
Синтаксис и читаемость	4	5	4	3	4	4
Производительность	3	3	4	2	4	4
Экосистема и сообщество	5	5	4	5	4	5
Масштабируемость	5	4	5	4	5	5
Удобство разработки	5	4	5	5	4	4
Безопасность	4	5	2	2	4	5
Веб-разработка	5	5	5	5	3	5
Сумма оценок	31	31	29	26	28	32

Исходя из оценок, полученных по каждому критерию, можно сделать следующие вывод, «C#» получил наивысшую суммарную оценку «31», что указывает на его превосходство по сравнению с другими языками. Из этого следует выбор данного языка программирования для разработки мобильного приложения.

Для выбора фреймворка проведем анализ самых популярных из них в виде таблицы 2 и определим какой из них следует выбрать для создания клиентской части подсистемы.

Таблица 2 – Анализ фреймворков

Критерии	**Xamarin**	**Flutter**	**React Native**	**.NET MAUI**
Популярность	5	5	3	5
Сообщество	5	5	3	5
Документация	4	4	3	5
Производительность	4	3	4	5
Гибкость	4	3	4	5
Обучаемость	4	3	4	4
Экосистема	4	4	2	4
Всего	30	27	23	33

Исходя из таблицы 2, фреймворк «.NET MAUI» имеет наивысшую сумму баллов, равную 33-м. Таким образом, для создания подсистемы следует выбрать именно «.NET MAUI».

В качестве СУБД для работы с базой данной системы были рассмотрены следующие системы: «PostgreeSQL», «Microsoft SQL» «Server», «SQLite», таблица 3.

Таблица 3 – Анализ СУБД

Критерии	**PostgreSQL**	**MS SQL Server**	**SQLite**
Функциональность	5	4	5
Производительность	4	5	5
Масштабируемость	4	5	4
Сообщество	5	4	4
Поддержка	5	4	5
Стоимость	4	3	5
Всего	27	25	33

Исходя из анализа очевидным выбором СУБД для разработки подсистемы будет «SQLite», также дополнительным критерием выбора к данной СУБД является его возможность к созданию локальной БД.

2.2 Разработка концепции архитектуры построения и платформы реализации информационных систем

UML (Unified Modeling Language) представляет собой стандартизированный язык моделирования, широко используемый для разработки программных продуктов. UML обеспечивает набор диаграмм, которые служат для визуализации и описания различных аспектов системы. Ниже перечислены основные типы диаграмм UML, часто применяемые в процессе проектирования программных продуктов:

– диаграмма вариантов использования (Use Case Diagram). Данная диаграмма помогает описать функциональность системы с точки зрения ее актеров, т.е. пользователей. Она иллюстрирует взаимодействие между актерами и системой, а также демонстрирует различные сценарии использования и их связи с функциональностью системы;

– диаграмма классов (Class Diagram). Диаграмма классов отображает структуру системы, представляя классы, их атрибуты, методы и связи между ними. Она помогает описать структуру данных и отношения между классами, а также иерархию наследования;

– диаграмма последовательности (Sequence Diagram). Диаграмма последовательности показывает взаимодействие объектов в определенной последовательности. Она демонстрирует, как объекты обмениваются сообщениями друг с другом в рамках определенного сценария или функции системы;

– диаграмма состояний (State Diagram). Диаграмма состояний описывает поведение объекта или системы в различных состояниях и переходах между ними. Она помогает моделировать дискретные состояния и условия, которые влияют на поведение системы в различных ситуациях;

– диаграмма компонентов (Component Diagram). Диаграмма

компонентов отображает физическую структуру системы и ее компоненты, такие как библиотеки, модули или сервисы, а также связи между ними. Она помогает понять архитектуру системы и компонентное разделение.

Приведенные выше диаграммы UML представляют лишь некоторые из основных типов, и также существуют другие типы диаграмм, такие как диаграмма развертывания, диаграмма активности и диаграмма объектов. Путем комбинирования различных диаграмм UML разработчики могут создавать более полные и понятные модели системы, которые помогают в процессе проектирования программных продуктов. Все диаграммы будут выполнены в ПО «ArgoUML».

Начнем проектирование системы с диаграммы прецедентов, указанной на рисунке 4, определим основные действующие лица подсистемы и варианты использования.

Рисунок 4 – Диаграмма прецедентов

В качестве действующих лиц (актеры) выступают:

– пользователь (редактирует и записывает данные о комплектующих автомобиля, заполняет и изменяет профиль и информацию об автомобиле);

– сотрудник СТО (получает данные о прохождении технического обслуживания автомобиля и использует ее для осуществления сервисного обслуживания);

2.3 Структура информационных систем, состав функциональных и обеспечивающих подсистем

Диаграмма классов является средством моделирования, которое позволяет определить типы классов в системе и статические связи между ними. Она также предоставляет возможность отобразить атрибуты классов, операции классов и ограничения, применяемые к связям между классами.

В процессе разработки системы, где требуется взаимодействие с базой данных, рекомендуется принять решение о выборе подхода, который предусматривает использование ORM (Object-Relational Mapping) для взаимодействия с базой данных. Использование ORM упрощает процесс разработки, поскольку классы в системе отражают сущности, представленные в базе данных. На рисунке 5 представлена диаграмма классов, иллюстрирующая сущности системы.

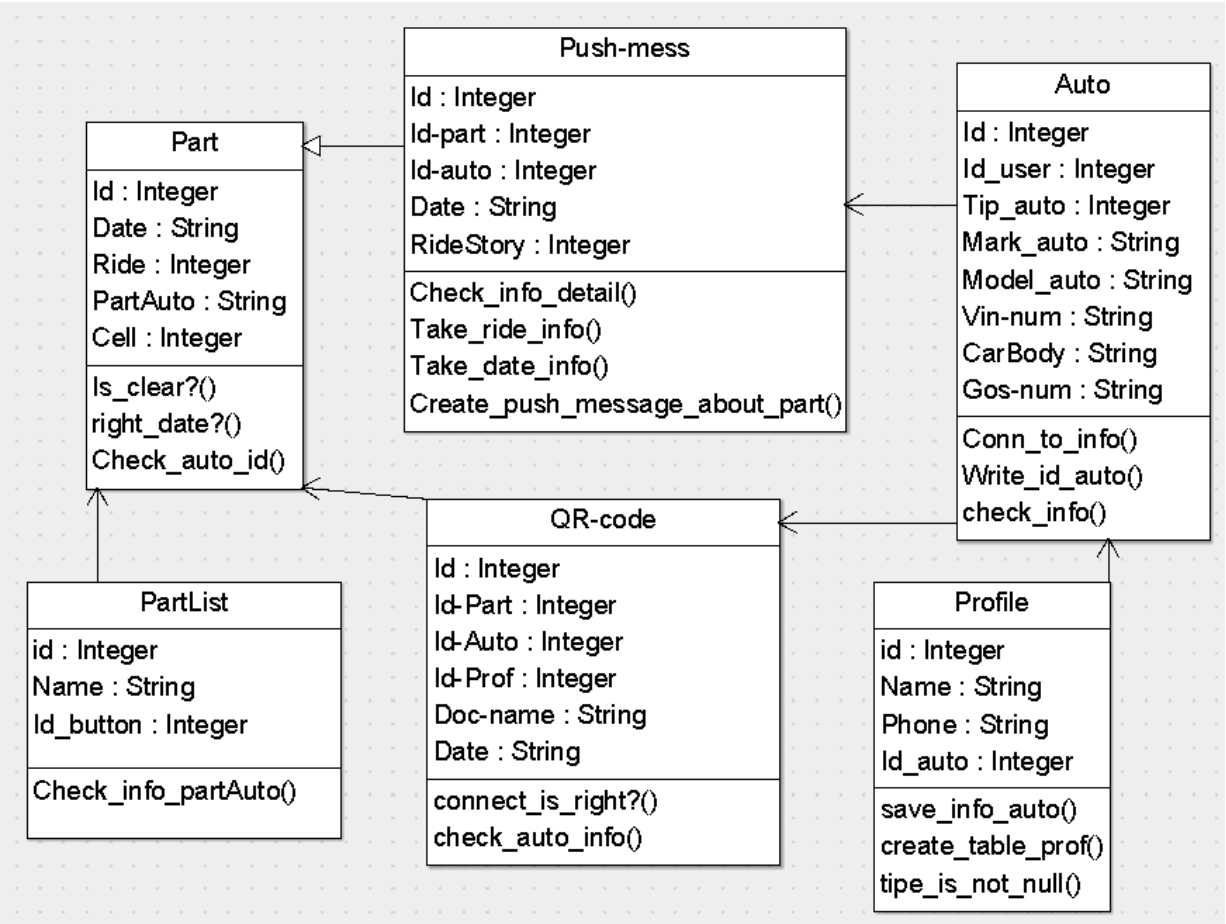

Рисунок 5 – Диаграмма классов подсистемы

Каталог «app» в организации компонентов приложения обладает подкаталогами, которые содержат представления «views» и библиотеки разных платформ «Platforms», ресурсы приложения «Resources» и базу данных «db». Он является ключевым каталогом приложения.

Подкаталог «app/Platforms» содержит подкаталоги поддерживаемых операционных систем, включая настройки стиля, прав доступа, используемых настроек и разрешений.

Подкаталог «app/Resources» содержит все используемые ресурсы в приложении, включая изображения, иконки, шрифты, персональные настройки стилей и другие. Данный каталог служит хранилищем ресурсов для визуализации приложения.

Подкаталог «app/Views» содержит формы, используемого приложения. Здесь происходит описание, редактирование и взаимодействие с визуальной состовляющей приложения

Подкаталог «app/db» обычно содержит скрипты и инструменты для управле-ния реляционной базой данных, используемой приложением. Здесь можно создавать, изменять и управлять структурой и содержимым базы данных, используя объекты моделей приложения. Этот каталог играет важную роль в поддержке и развитии приложения, связанной с базой данных.

Файловая структура приложения представлена на рисунке 6.

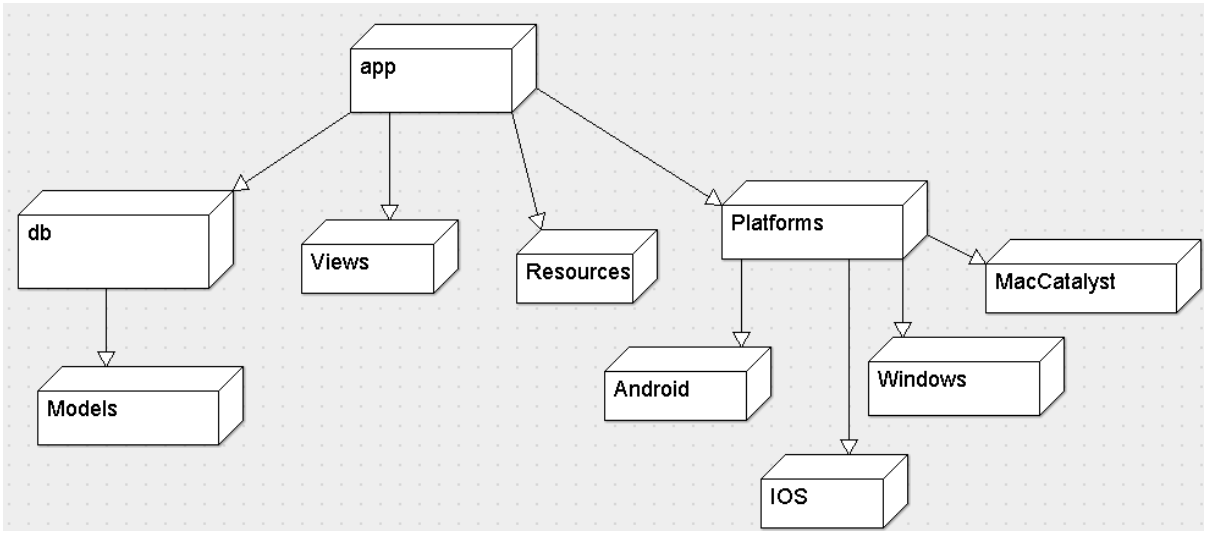

Рисунок 6 – Файловая структура приложения

Диаграмма развертывания – это тип диаграммы UML, которая иллюстрирует архитектуру выполнения системы, включая такие узлы, как аппаратные или программные среды выполнения, а также промежуточное программное обеспечение, которое их соединяет (рис. 7).

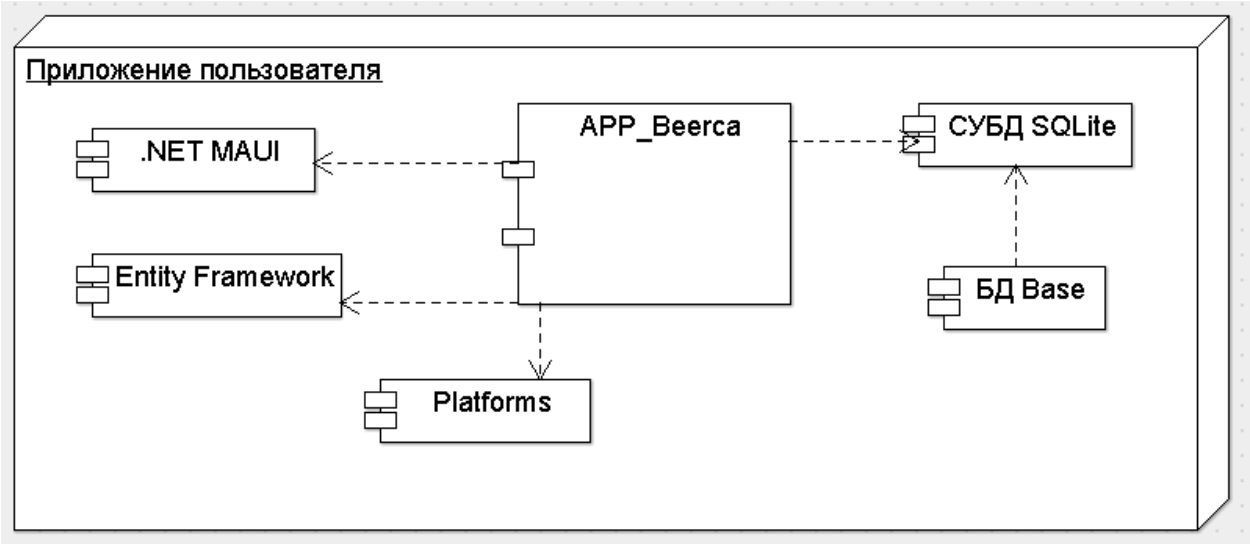

Рисунок 7 – Диаграмма развертывания

В архитектуре толстого клиента клиентские устройства имеют более высокую вычислительную мощность и хранят значительное количество данных и программного обеспечения. Они выполняют значительную часть вычислений и обработки данных непосредственно на устройствах клиента.

Преимущества толстого клиента заключаются в распределении вычислительной нагрузки между клиентскими устройствами и серверами. Это может привести к улучшению производительности системы, поскольку некоторые вычисления и обработка данных могут выполняться локально на клиентских устройствах, не требуя постоянного обращения к серверу. Кроме того, такая архитектура позволяет более гибко управлять данными и программным обеспечением на клиентских устройствах, так как они могут хранить и обрабатывать больший объем информации независимо от сервера.

Однако у толстого клиента также есть свои недостатки. Поскольку клиентские устройства имеют больше вычислительных возможностей и хранят больше данных и программного обеспечения, они могут быть более дорогими

и сложными в управлении по сравнению с тонким клиентом. Кроме того, возникает проблема обновления программного обеспечения на клиентских устройствах, поскольку каждое устройство должно быть обновлено отдельно. Безопасность также может быть более сложной, поскольку данные хранятся и обрабатываются на клиентских устройствах, что может представлять риск в случае утери или кражи устройства.

2.4 Формулировка требований к программному продукту

Исходя из анализа существующих решений можно однозначно заключить, что специфика разработки мобильного приложения для сервисного обслуживания автомобиля является актуальной. Можно выделить следующие требования к такому программному продукту:

– русскоязычный интерфейс;

– работа на компьютерах с ОС Windows 10 и новее;

– кроссплатформенность;

– возможности записи данных сервисного обслуживания;

– возможность ведения учета нескольких автомобилей;

– безопасное хранение и синхронизация данных;

– оптимизированная работа приложения;

– наличие удобного и понятного интерфейса.

Анализ существующих решений показывает, что разработка мобильного приложения для сервисного обслуживания автомобиля является актуальной задачей. Для такого программного продукта можно выделить несколько важных требований. Прежде всего, приложение должно иметь русскоязычный интерфейс, т. к. данное проектное решение нацелено на российскую аудиторию. Оно также должно работать на компьютерах с операционной системой Windows 10 и новее, чтобы быть совместимым с широким спектром устройств.

Кроссплатформенность является ещё одним важным требованием, чтобы приложение могло быть установлено и использовано на различных операционных системах, таких как Windows, macOS, а также на мобильных

устройствах под управлением iOS или Android.

Функциональность записи данных сервисного обслуживания и возможность ведения учета нескольких автомобилей являются неотъемлемыми частями такого приложения. Пользователям должна быть предоставлена возможность удобно отслеживать информацию о проведенных обслуживаниях и ремонтах своих автомобилей.

Безопасное хранение и синхронизация данных – это критически важный аспект для такого приложения. Данные пользователей, включая информацию о техническом состоянии автомобилей и истории обслуживания, должны быть надежно защищены и доступны только пользователю.

Оптимизированная работа приложения и наличие удобного и понятного интерфейса играют важную роль в обеспечении удовлетворения пользователей. Приложение должно быть отзывчивым, быстрым и легким в использовании, чтобы предоставить приятный и эффективный пользовательский опыт.

Учитывая эти требования, разработка мобильного приложения для сервисного обслуживания автомобиля представляет значимую возможность внедрения полезного и востребованного программного продукта на рынке.

2.5 Выбор и обоснование модели жизненного цикла

Существует несколько методик разработки программного обеспечения, каждая из которых предлагает свой подход к управлению процессом разработки. Одной из наиболее популярных методик является «Spiral Model», или спиральная модель, которая объединяет элементы каскадной модели и итеративного подхода. На рисунке 8 представлена визуализация спиральной модели. Эта методика предлагает циклический процесс разработки, состоящий из четырех основных фаз: определение целей и альтернатив, оценка рисков, разработка и тестирование.

В рамках спиральной модели каждый цикл разработки начинается с создания небольшого прототипа или версии продукта, которая затем анализируется и оценивается перед переходом к следующему циклу. Такой итеративный

подход позволяет постепенно уточнять и совершенствовать продукт на основе полученных результатов и обратной связи.

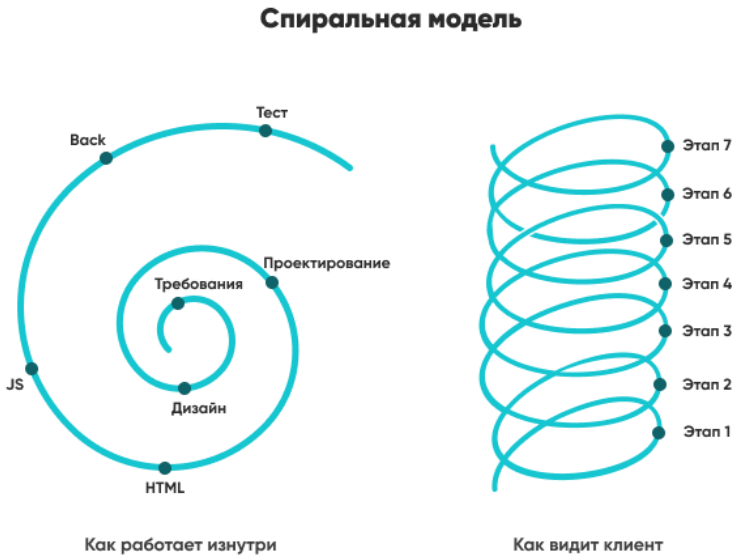

Рисунок 8 – Спиральная модель

Еще одной распространенной методикой разработки является каскадная модель, изображенная на рисунке 9. Каскадная модель представляет собой линейный подход к разработке программного обеспечения, где каждая фаза проекта выполняется последовательно. Этапы разработки включают сбор требований, проектирование, разработку, тестирование и развертывание. Каждая фаза завершается перед переходом к следующей, и внесение изменений в требования после начала разработки может быть затруднительным.

Каскадная модель предоставляет структурированный подход к разработке, который может быть особенно полезен при проектах с жесткими требованиями и четко определенным объемом работ. Она обеспечивает последовательность и четкость в выполнении фаз проекта, что может способствовать лучшему контролю над процессом разработки. Однако, ограничения на изменение требований после начала разработки могут снижать гибкость и адаптивность модели к изменяющимся потребностям проекта.

Использование определенной методики разработки программного

обеспечения зависит от особенностей проекта, его требований и предпочтений команды разработчиков. Каждая методика имеет свои преимущества и недостатки, и выбор конкретной методики требует внимательного анализа и согласования с командой разработчиков и заказчиком.

Рисунок 9 – Каскадная модель

Агиловые методологии, такие как Scrum и Kanban, представляют собой итеративные и инкрементальные подходы к разработке программного обеспечения. Они основываются на принципах гибкости, коммуникации и быстрого реагирования на изменения требований. Эти методологии используют короткие циклы, называемые спринтами, в течение которых команда разработчиков занимается выполнением конкретных задач в ограниченный промежуток времени.

DevOps является философией и методологией, которая объединяет разработку и операции в единый процесс. Рисунок 10 иллюстрирует эту концепцию. DevOps стремится к автоматизации и сокращению разрыва между разработчиками и операционными командами. Это позволяет доставлять программное обеспечение в эксплуатацию быстрее и надежнее.

После проведения анализа наиболее распространенных моделей

жизненного цикла разработки программного продукта было принято решение остановиться на каскадной модели. Выбор данной модели обоснован ее способностью эффективно управлять изменениями требований к продукту, а также обеспечивать оперативную генерацию результатов на каждой итерации и оценку работы программного продукта относительно прототипа.

Рисунок 10 – DevOps методология

Каскадная модель также является подходящим вариантом для расширения функциональности, так как она предоставляет гибкость внесения изменений в процесс разработки новых версий прототипа. Это особенно важно в области программной разработки, где требования и технические аспекты могут изменяться со временем.

2.6 Требования к информационной безопасности для приложений «.NET MAUI»

Мобильные приложения, разработанные с использованием .NET MAUI (Multi-platform App UI), могут быть подвержены различным уязвимостям, которые характерны для программной разработки в целом.

Неправильное обращение к данным, атаки типа межсайтового скриптинга (XSS), инъекции SQL и другие виды атак могут возникать из-за ошибок в коде приложения. Некорректная обработка пользовательского ввода,

отсутствие проверки данных и неправильная реализация механизмов авторизации и аутентификации могут привести к компрометации данных или их злоупотреблению.

Если не используется надежная система аутентификации и авторизации, злоумышленники могут получить несанкционированный доступ к функциональности приложения или к пользовательским данным. Недостаточно надежные пароли, отсутствие механизмов двухфакторной аутентификации или уязвимости в процессе аутентификации могут привести к утечке данных или несанкционированному использованию аккаунтов.

Уязвимости хранения данных - если данные, такие как пароли, личная информация пользователей или другие конфиденциальные данные, не хранятся должным образом, они могут быть украдены или скомпрометированы. Несанкционированный доступ к локальным базам данных, хранение данных в незашифрованном виде на устройстве или недостаточная защита данных в оперативной памяти могут привести к утечке конфиденциальной информации.

Уязвимости в сетевом взаимодействии - мобильные приложения часто взаимодействуют с удаленными серверами посредством сети. Небезопасная передача данных, открытые или недостаточно защищенные API, атаки на сетевой уровень или подмена данных могут привести к компрометации информации или злоупотреблению.

Уязвимости в сторонних компонентах - мобильные приложения могут использовать сторонние библиотеки, фреймворки или плагины, которые могут содержать уязвимости. Уязвимости во внешних компонентах могут быть использованы злоумышленниками для атаки на приложение или устройство пользователя.

3 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

Для создания ИС необходимо выполнить ряд шагов разработки:

– сформулировать требования;

– выбрать инструменты и средства разработки;

– описать модель ИС;

– спроектировать структуру модулей и классов;

– спроектировать базу данных;

– спроектировать пользовательский интерфейс;

– реализовать ИС на основе выбранных компонентов;

– протестировать ИС.

3.1 Формулировка требований к информационной системе

К разрабатываемому программному продукту предъявляются следующие требования:

– совместимость с Windows, macOS, Android, IOS;

– эффективное выполнение целевых функций программы;

– отслеживание изменений состояния сканируемых устройств;

– возможность экспортирования данных в различные форматы;

– простой, интуитивно понятный пользовательский интерфейс.

В результате анализа требований к разрабатываемому программному продукту, можно сделать следующий общий вывод. Для успешной разработки программного продукта необходимо обеспечить его совместимость с платформами Windows, macOS, Android и iOS, чтобы охватить широкий круг пользователей. Важно, чтобы программа эффективно выполняла свои целевые функции, обеспечивала надежное отслеживание изменений состояния сканируемых устройств и предоставляла возможность экспортировать данные в различные форматы. Кроме того, для удобства пользователей необходимо предусмотреть простой и интуитивно понятный пользовательский интерфейс. Удовлетворение этих требований способствует разработке качественного

программного продукта, который будет отвечать потребностям пользователей и обеспечивать удобство и эффективность его использования.

3.2 Структура информационной системы

3.2.1 Лингвистическое обеспечение

Приложение будет разработано на языке C#, как наиболее универсальный и подходящий требованиям заказчика.

Пользовательский интерфейс и интерфейс приложения будет выполнен на русском языке, за исключением указания комплектующих (Могут присутствовать англоязычные названия).

3.2.2 Информационное обеспечение

Для обеспечения корректной работы программы необходимо подключение к сети интернет, доступ к контактам, для связи с сервисным центром и доступ к геолокации, для возможности записи на обслуживание.

Для получения информации о необходимых запчастях и затратах будет использован VIN номер автомобиля.

В окне пользователя будет отображена основная информация о пользователе, включая контактные данные.

В окне автомобиля будет отображена основная информация об автомобиле.

3.2.3 Программное обеспечение

Для разработки приложения будет использоваться Microsoft Visual Studio 2022 года с дополнением Xamarin.

База Данных будет разработана на PostgreSQL.

3.2.4 Техническое обеспечение

Приложение будет работать на различных ОС мобильных телефонах и должно включать в себя данные минимальные требования версий:

– Android 7.0 и выше;

– IOS 11.0 и выше;

– Symbian OS 7.0.4 и выше;

– BlackBerry OS 5.0 и выше;

– Windows Embedded CE 6.0 и выше.

Для поддержания клиентской базы данных будет использован сервер, с рекомендуемыми требованиями:

– Дисковое пространство, организованное в формате Raid 10, объемом не менее 512 гб;

– Доступ к сети интернет и пропускной мощностью канала не менее 1000/1000 (Mbps).

3.2.5 Функциональные подсистемы

Подсистема «QR - Код». В данной функциональной подсистеме будет доступно сканирование QR – кода для последующего вывода отчета по всему техническому обслуживанию, с указанием дат, затрат, месте проведения, и замененного комплектующего.

Подсистема «Обслуживание». В данной функциональной подсистеме будет хранится информация по определенной обслуживаемой части автомобиля и занесение информации пользователем.

Подсистема «Настройки». В данной функциональной подсистеме будут находиться различные настройки программы, например изменения пользовательской темы, включение режима для слабовидящих и т.д.

Подсистема «Техника». В данной подсистеме находится информация о каждой занесенной техники пользователем с указанием типа, модели, регистрационного номера, данных об обслуживании и т.д.

Подсистема «Пользователи». В данной подсистеме расположена информация о пользователях и дальнейшее использование данных для записи на обслуживание, учет техники, составление отчетов и т.д.

3.3 Выбор инструментов и средств разработки

3.3.1 Выбор языка программирования

Использование языка программирования C# при разработке приложения для инвентаризации компьютеров демонстрирует ряд значительных

преимуществ.

Во-первых, C# является высокоуровневым языком программирования, обладающим широким набором инструментов и библиотек, которые способствуют удобству и продуктивности разработки. Синтаксис C# оказывается интуитивно понятным и читабельным, что обеспечивает быструю и эффективную разработку приложений.

Во-вторых, C# тесно интегрирован с платформой .NET, являясь ее основным языком программирования. Это позволяет разработчикам использовать обширный набор классов, библиотек и инструментов, предоставляемых платформой .NET, что обеспечивает широкие возможности в области работы с сетью, базами данных, пользовательским интерфейсом и другими компонентами, важными для создания приложения для инвентаризации.

В-третьих, язык C# обладает встроенной поддержкой асинхронного программирования, что способствует эффективной работе с асинхронными операциями, такими как получение данных из сети или выполнение запросов к базе данных. Это особенно полезно в контексте разработки приложений для инвентаризации, где может быть необходимо выполнение долгих операций сетевого сканирования или сбора информации о компьютерах.

В-четвертых, приложения, разработанные на C#, обладают масштабируемостью и переносимостью. Они могут легко масштабироваться и переноситься на различные платформы и операционные системы, включая Windows, Linux и macOS. Это обеспечивает гибкость и универсальность при создании приложений для инвентаризации, которые могут быть успешно развернуты в различных средах.

В целом, использование языка C# при разработке приложения для инвентаризации предоставляет разработчикам мощные инструменты, простой синтаксис и возможность эффективной работы с сетью, базами данных и асинхронными операциями, что способствует созданию надежных и функциональных приложений данного типа.

3.3.2 Выбор СУБД

Для управления базами данных в разрабатываемом решении была принята решающая роль Система Управления Базами Данных (СУБД) SQLite. Применение языка программирования C# и платформы .NET позволяет работать с базами данных в данном решении в виде классов, что устраняет необходимость написания SQL-запросов. Вместо этого разработчики могут использовать библиотеку LINQ, предоставляющую удобный интерфейс для работы с обобщениями. Такой подход сокращает время и усилия, затрачиваемые на разработку и отладку SQL-запросов.

Кроме того, использование СУБД SQLite вместе с языком C# и платформой .NET способствует ускорению процессов записи и чтения данных в программе. Так как SQLite не требует наличия отдельного сервера для своей работы, это позволяет упростить и ускорить доступ к базе данных, исключая необходимость настройки и поддержки отдельного серверного окружения.

Таким образом, выбор SQLite в сочетании с языком C# и платформой .NET предоставляет эффективные средства работы с базами данных, повышает производительность программного решения и упрощает процесс разработки и обслуживания базы данных.

3.3.3 Выбор графической подсистемы

При работе с языком программирования C# и платформой .NET, доступны различные графические подсистемы, такие как Windows Forms, WPF, UWP и MAUI. Однако, для данного программного продукта не рекомендуется использовать UWP и MAUI, так как эти варианты не поддерживают работу в Windows 7, которая является требованием для данного проекта из-за наличия компьютеров с данной операционной системой в организации.

Существуют различия между Windows Forms и Windows Presentation Foundation (WPF).

WPF предлагает современные и гибкие возможности визуализации, включая векторную графику, анимацию, привязку данных, стили и шаблоны.

Это позволяет создавать более богатые и интерактивные пользовательские интерфейсы. В то время как Windows Forms имеет более простую и ограниченную графическую модель.

.NET MAUI (Multi-platform App UI) – это фреймворк для разработки кроссплатформенных мобильных и настольных приложений, который объединяет преимущества Windows Forms, WPF и Xamarin.Forms. Он предоставляет гибкую компоновку элементов с использованием панелей и сеток для точного позиционирования, подобно Windows Forms, а также предлагает мощные возможности по созданию стилей и тем, подобно WPF.

.NET MAUI предоставляет больше возможностей для кроссплатформенной разработки приложений, в то время как WPF ориентирован на создание приложений для операционной системы Windows.

Однако, для данного проекта предпочтительным графическим подсистемой будет .NET MAUI. Он обеспечивает интуитивно понятный интерфейс и оптимальную работу в течение длительного времени, а также предоставляет привязку данных, что помогает сократить объем создаваемого программного кода.

3.3.4 Выбор интегрированной среды разработки

При анализе двух популярных решений для разработки проектов на .NET – Visual Studio от Microsoft и Rider от JetBrains, были выявлены их отличительные особенности. Visual Studio обладает рядом преимуществ, таких как расширенный IntelliSense, что обеспечивает интеллектуальную подсказку и предсказание кода, улучшая производительность и качество конечного продукта. Кроме того, Visual Studio предоставляет бесплатную Community-версию, что позволяет доступно использовать инструментарий разработки. Также, Visual Studio поддерживает работу с проектами .NET MAUI, что предоставляет расширенные возможности для работы с графической подсистемой и упрощает разработку интерфейса с использованием XAML.

С другой стороны, Rider отличается более строгим контролем над

именованием переменных, методов, полей, структур и классов, что помогает в соблюдении стандартов кодирования и уменьшении возможности ошибок. Однако, у Rider все версии распространяются платно, за исключением версии для студентов, что требует подтверждения статуса студента.

Исходя из проведенного анализа, для разработки выбрано решение .NET MAUI for Visual Studio, так как оно обладает расширенными функциональными возможностями, бесплатной доступной версией и удобством работы с интерфейсом.

3.4 Разработка структуры базы данных

3.4.1 Описание инфологической структуры

Определён такой набор сущностей:

– профиль (хранит информацию о пользователе);

– запчасти (хранит информацию о запчастях автомобиля);

– список_комплектующих (хранит информацию для перехода к нужным комплектующим);

– QR-код (хранит информацию о формировании формы посредством сканирования QR-кода);

– авто (хранит информацию о автомобиле пользователя);

– push-уведомление (хранит информацию, необходимую для формирования push-уведомления);

Таблица 4 – Сущность «профиль»

Название атрибута	**Описание**	**Тип данных**	**Диапазон значений**	**Пример атрибута**
Идентифи-катор	Однозначно идентифицирует экземпляр сущности в масштабе таблицы	Числовой	0-65535	40
Имя пользователя	Имя, указанное пользователем	Строка	–	Алексей
Номер телефона	Контактный номер телефона пользователя	Строка	–	89294567887

Таблица 5 – Сущность «запчасти»

Название атрибута	**Описание**	**Тип данных**	**Диапазон значений**	**Пример атрибута**
Идентифи-катор	Однозначно идентифицирует экземпляр сущности в масштабе таблицы	Числовой	w	40
Дата_Замены	Дата проведенной замены	Дата	01.01.2000 00:00 – 31.12.3000 23:59	06.06.2023
Пробег	Пробег на момент замены	Число	–	96000
Название_Детали	Наименование используемой запчасти	Строка	–	Liqui Moly
Стоимость	Затраты на замену	Число		5600

Таблица 6 – Сущность «Список_комплектующих»

Название атрибута	**Описание**	**Тип данных**	**Диапазон значений**	**Пример атрибута**
Идентифи-катор	Однозначно идентифицирует экземпляр сущности в масштабе таблицы	Числовой	0-65535	40
Название_Детали	Дата создания документа	Строка	–	Стеклоомывающая жидкость
Идентификатор_кнопки	Наименование документа	Строка	–	Kldjsa-dsak93812-sk

Таблица 7 – Сущность «авто»

Название атрибута	**Описание**	**Тип данных**	**Диапазон значений**	**Пример атрибута**
1	2	3	4	5
Идентифи-катор	Однозначно идентифицирует экземпляр сущности в масштабе таблицы	Числовой	0-65535	40
Тип_Авто	Тип транспорта	Строка	–	Электрокар
Марка	Марка автомобиля	Строка	–	Nissan
Модель	Модель автомобиля	Строка	–	Note
VIN-номер	VIN-номер автомобиля	Строка	17	XTA21124070445066

Продолжение таблицы 7

Госномер	Государственный номер автомобиля	Строка	9	О999ОО28

Таблица 8 – Сущность «QR-код»

Название атрибута	**Описание**	**Тип данных**	**Диапазон значений**	**Пример атрибута**
Идентифи-катор	Однозначно идентифицирует экземпляр сущности в масштабе таблицы	Числовой	0-65535	40
Дата_создания	Дата создания документа	Дата	01.01.2000 00:00 – 31.12.3000 23:59	06.12.2023
Название_документа	Название созданного документа с использование идентификационного номера	Строка	–	Отчет по техническому обслуживанию автомобиля Nissan Note №46012

Таблица 9 – Сущность «Push-уведомления»

Название атрибута	**Описание**	**Тип данных**	**Диапазон значений**	**Пример атрибута**
Идентифи-катор	Однозначно идентифицирует экземпляр сущности в масштабе таблицы	Числовой	0-65535	40
Пройдено	Записанное расстояние после замены	Числовой	–	6789
Время	Дата, по достижению которой, считается, что нужно обратить внимание на деталь	Дата	01.01.2000 00:00 – 31.12.3000 23:59	14.08.2024

Заключительном этапом при инфологическом проектировании является создания модели сущность-связь в нотации Мартина, Чена или IDEF1X. Для построения модели БД в данной курсовой работе была выбрана нотация Мартина, ER-диаграмма показана на рисунке 11.

Инфологическая схема базы данных будет реализована с использование свободно распространяемого программного обеспечения «draw.io».

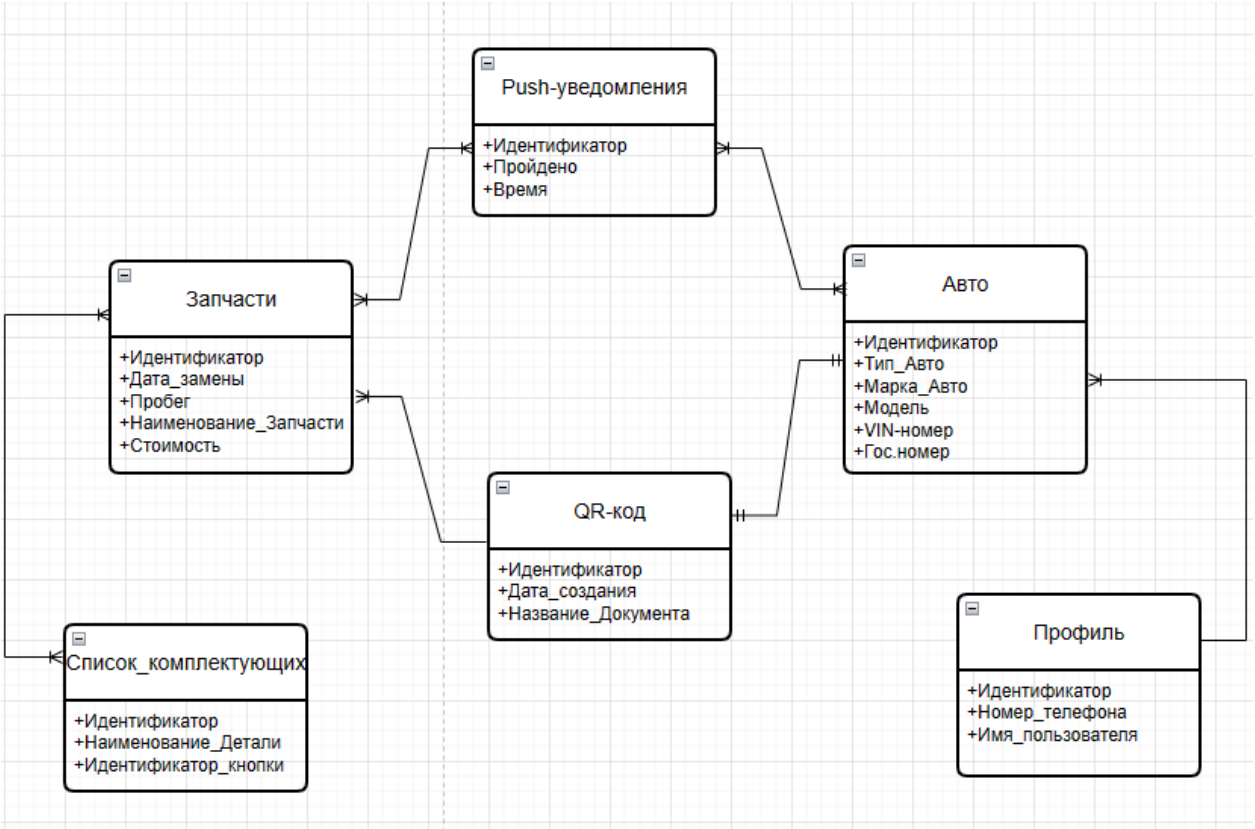

Рисунок 11 Инфологическая модель

3.4.2 Описание логической структуры

Логическое проектирование базы данных (ЛПБД) — это этап разработки базы данных, на котором определяются ее структура и связи между данными. Целью логического проектирования является создание схемы базы данных, которая будет эффективно хранить и обрабатывать информацию, отражая требования и бизнес-правила организации.

Связь «QR-код» – «Запчасти». Рассмотрим две сущности: «QR-код» и «Запчасти». Между ними установлена связь типа «один-ко-многим». Один документ в QR-коде может содержать информацию по нескольким запчастям, связь показана на рисунке 12. Исходя из данной информации добавим внешний ключ к сущности «Запчасти», так как он является дочерним по отношению к сущности «QR-код».

Рисунок 12 – Отношение «QR-код – Запчасти»

Рассмотрим две сущности: «Список_комплектующих» и «Запчасти». Между ними установлена связь типа «многие-ко-многим». Одна и та же запчасть может быть использована в нескольких автомобилях, и в то же время у одного комплектующего может быть несколько запчастей, связь показана на рисунке 13.

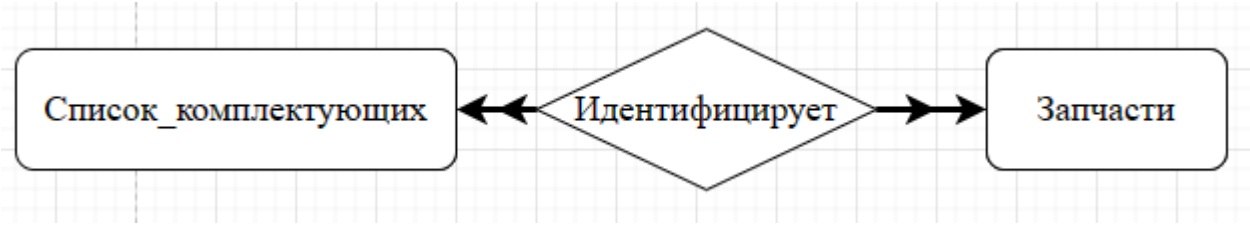

Рисунок 13 – Отношение «Список_комплектующих – Запчасти»

Далее рассмотрим две сущности: «Push-уведомления» и «Запчасти». Между ними установлена связь типа «один- к-одному». Одному виду запчасти может соответствовать одному уведомлению, связь представлена на рисунке 14.

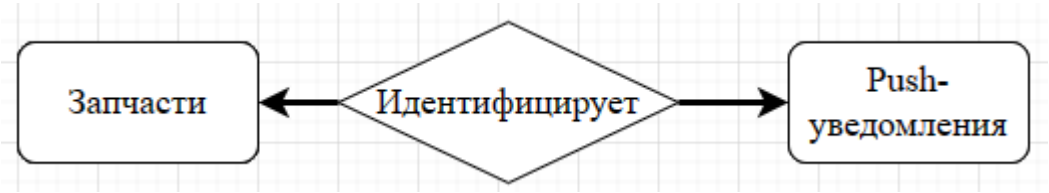

Рисунок 14 – Отношение «Push-уведомления – Запчасти»

Связь «Профиль» – «Авто». Рассмотрим две сущности: «Профиль» и «Авто». Между ними установлена связь типа «один-ко-многим». Одному пользователю могут принадлежать несколько автомобилей, в то время как каждый автомобиль принадлежит одному конкретному пользователю, связь показана на рисунке 15.

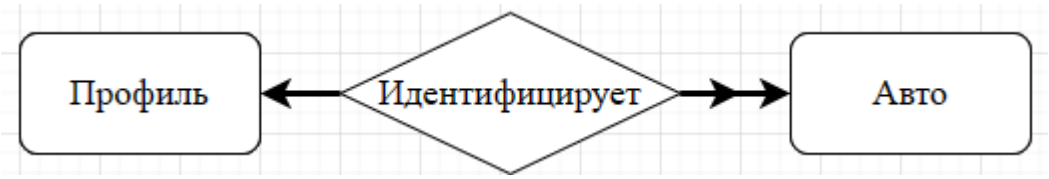

Рисунок 15 – Отношение «Профиль – Авто»

Связь «Push-уведомления» – «Авто». Рассмотрим две сущности: «Push-уведомления» и «Авто». Между ними установлена связь типа «один-ко-многим». Одному автомобилю может принадлежать несколько уведомлений в то время, как уведомления формируются по конкретному автомобилю, связь показана на рисунке 16.

Рисунок 16 – Отношение «Push-уведомления – Авто»

Связь «QR-код» – «Авто». Рассмотрим две сущности: «Push-уведомления» и «Авто». Между ними установлена связь типа «один-к-одному». Одному автомобилю может принадлежать один отчет, связь показана на рисунке 17. Добавим внешний ключ в сущность «Авто», так как он является дочерним по отношению к сущности «QR-код».

Рисунок 17 – Отношение «QR-код – Авто»

3.4.3 Реализация физической структуры

На основании составленных таблицы атрибутов были установлены необходимые связи между сущностями, разрешена связь «многие-ко-многим», а также определены необходимые типы данных. Диаграмма, представляющая связи между сущностями, теперь может быть использована для физического

проектирования базы данных. Физическое проектирование БД представляет собой процесс создания физической структуры БД на основе логического моделирования. В таблицах 18 – 23 приведены физические представления атрибутов сущностей.

Физическое представление атрибутов включает в себя выбор оптимальных типов данных для хранения значений атрибутов, определение ограничений целостности и индексирование данных для обеспечения эффективного доступа и запросов к БД.

Таблица 18 – Физическая структура данных сущности «профиль»

Название атрибута	**Тип данных**	**Условия**	**Формат данных**	**Индексация**
Идентифи-катор	Число	>0	bigint	Primary key
Имя пользователя	Строка	Не пустое	String	–
Номер телефона	Строка	Не пустое	String	–

Таблица 19 – Физическая структура данных сущности «запчасти»

Название атрибута	**Тип данных**	**Условия**	**Формат данных**	**Индексация**
Идентифи-катор	Число	>0	bigint	Primary key
Дата_Замены	Дата	> 01.01.1940	Date	–
Пробег	Строка	Не пустое	String	–
Название_Детали	Строка	Не пустое	String	–
Стоимость	Строка	Не пустое	String	–

Таблица 20 – Физическая структура данных сущности «Список_комплектующих»

Название атрибута	**Тип данных**	**Условия**	**Формат данных**	**Индексация**
Идентифи-катор	Число	>0	bigint	Primary key
Название_Детали	Строка	Не пустое	String	–
Идентификатор_кнопки	Строка	Не пустое	String	–

Таблица 21 – Физическая структура данных сущности «авто»

Название атрибута	**Тип данных**	**Условия**	**Формат данных**	**Индексация**
1	2	3	4	5
Идентифи-катор	Число	>0	bigint	Primary key

Продолжение таблицы 21

1	2	3	4	5
Тип_Авто	Строка	Не пустое	String	–
Марка	Строка	Не пустое	String	–
Модель	Строка	Не пустое	String	–
VIN-номер	Строка	Не пустое	String	–
Госномер	Строка	Не пустое	String	–

Таблица 22 – Физическая структура данных сущности «QR-код»

Название атрибута	**Тип данных**	**Условия**	**Формат данных**	**Индексация**
Идентифи-катор	Число	>0	bigint	Primary key
Дата_создания	Дата	> 01.01.1940	Date	–
Название_документа	Строка	Не пустое	String	–

Таблица 23 – Физическая структура данных сущности «Push-уведомления»

Название атрибута	**Тип данных**	**Условия**	**Формат данных**	**Индексация**
Идентифи-катор	Число	>0	bigint	Primary key
Пройдено	Строка	Не пустое	String	–
Время	Строка	Не пустое	String	–
Название_Детали	Строка	Не пустое	String	Foreign key

Завершим этап реализации физической структуры БД, путем построения схемы в формате «IDEF1X», схема показана на рисунке 18.

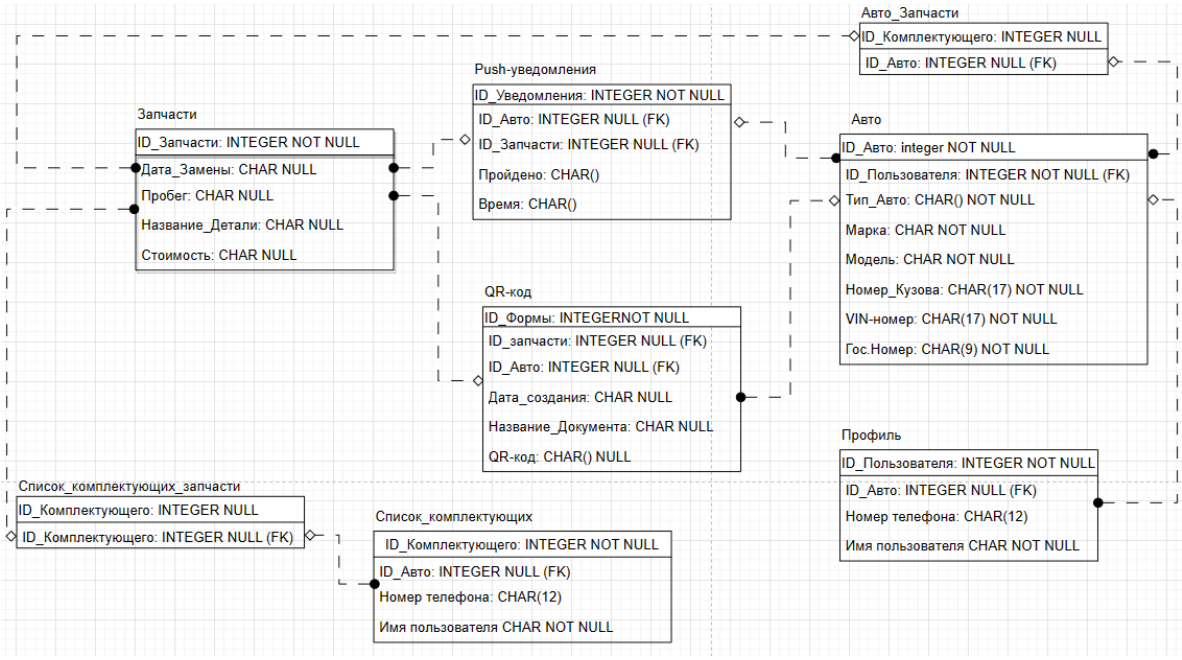

Рисунок 18 – Физическое проектирование

3.5 Реализация программного продукта

3.5.1 Обеспечение работы с комплектующими

Для записи информации по комплектующих в приложении используется библиотека sqlite-net-pcl. Она предоставляет удобный и простой в использовании API для выполнения операций чтения и записи данных в базу данных SQLite.

Библиотека sqlite-net-pcl представляет собой мощный инструмент для работы с базой данных SQLite в приложениях, разработанных на платформе .NET MAUI. Эта библиотека обеспечивает удобный интерфейс и предоставляет классы, позволяющие выполнять различные операции с записью информации.

Одним из ключевых классов в библиотеке sqlite-net-pcl является класс SQLiteConnection. Он предоставляет возможность установления соединения с базой данных SQLite и выполняет функции контекста для взаимодействия с базой данных. С помощью этого класса можно создавать, изменять и удалять таблицы, а также выполнять различные операции с данными.

Для работы с записью информации в базу данных SQLite в библиотеке sqlite-net-pcl предоставляются классы, которые обеспечивают удобную и эффективную работу с данными.

Класс TableMapping. Этот класс представляет собой отображение между типом данных объекта и таблицей в базе данных SQLite. Он содержит информацию о столбцах таблицы, их типах данных и соответствующих свойствах объекта. Класс TableMapping позволяет определить структуру таблицы и управлять операциями вставки, обновления и удаления данных.

Библиотека sqlite-net-pcl позволяет определить классы-модели данных, которые представляют сущности, хранящиеся в базе данных SQLite. Каждый класс-модель представляет таблицу в базе данных и содержит свойства, соответствующие столбцам таблицы. Библиотека автоматически выполняет маппинг между объектами класса-модели и таблицей базы данных.

Библиотека sqlite-net-pcl предоставляет классы, которые позволяют выполнять различные операции с данными, такие как вставка, обновление, удаление и выборка. Например, класс SQLiteCommand представляет SQL-запрос и позволяет выполнить операцию над данными в базе данных. Классы SQLiteQuery и SQLiteQueryAsync позволяют выполнять запросы и получать результаты выборки данных.

Использование классов и методов из библиотеки sqlite-net-pcl позволяет разработчикам легко и эффективно работать с записью информации в базу данных SQLite в приложениях .NET MAUI. Они предоставляют удобный интерфейс для создания таблиц, определения структуры данных, выполнения операций с данными и обеспечивают надежное хранение информации в базе данных.

3.5.2 Разработка системы работы push-уведомлений

Для реализации push-уведомлений был использован инструмент FCM (Firebase Cloud Messaging). FCM (Firebase Cloud Messaging) является сервисом, предоставляемым Firebase, который позволяет разработчикам отправлять push-уведомления на устройства пользователей. Он обеспечивает удобный способ взаимодействия с мобильными приложениями и облачными серверами, позволяя отправлять уведомления в режиме реального времени. FCM интегрируется с .NET MAUI с помощью специальной библиотеки, предоставляемой Firebase.

С использованием этой библиотеки разработчик может настроить своё приложение для получения push-уведомлений и обрабатывать их при поступлении. Основная идея FCM заключается в том, что разработчик может отправлять уведомления на конкретные устройства или группы устройств с помощью уникальных идентификаторов, известных как токены устройств. Приложение может получать эти токены и регистрировать их на сервере FCM для получения уведомлений.

Для обработки push-уведомлений в .NET MAUI разработчик может

создать сервисный класс, который будет подписываться на события получения уведомлений от FCM и выполнять необходимые действия в ответ на получение уведомления. Этот класс может быть связан с представлением, чтобы отображать полученные уведомления пользователю или выполнять другие действия при поступлении уведомления.

Включение использования Firebase Cloud Messaging (FCM) в .NET MAUI предоставляет эффективный способ реализации push-уведомлений в приложении. FCM обеспечивает удобный механизм взаимодействия между базой данных и мобильным устройством, позволяя доставлять уведомления в режиме реального времени.

В контексте составления документа, включение поддержки FCM в .NET MAUI представляет важный аспект разработки мобильного приложения для обслуживания автомобиля. Этот функционал позволит установить надежное и мгновенное соединение между сервером приложения и мобильным устройством пользователя. Таким образом, при возникновении событий, требующих внимания пользователя, приложение сможет отправлять push-уведомления в режиме реального времени, чтобы информировать пользователя о различных обновлениях, например, о предстоящей замене деталей автомобиля, необходимости прохождения технического обслуживания и других важных событиях.

Использование FCM в .NET MAUI значительно улучшает функциональность и практичность приложения для обслуживания автомобиля, обеспечивая надежный и эффективный механизм связи с мобильными устройствами пользователей, а также обеспечивая возможность мгновенной доставки уведомлений. Это значительно повышает удобство использования приложения и обеспечивает своевременную информацию для пользователей, помогая им поддерживать и контролировать состояние своего автомобиля.

3.5.3 Обеспечение отзывчивой работы программы

В языке C# для обеспечения асинхронной функциональности используются операторы async/await и класс Task. Класс Task представляет

асинхронную операцию, которая может быть выполнена в фоновом потоке или на пуле потоков. Этот класс принадлежит пространству имен System.Threading.Tasks и обладает удобными методами и свойствами для работы с асинхронными задачами. Ниже представлены основные аспекты, связанные с использованием класса Task:

– класс Task предоставляет статические методы для создания и запуска задач. Например, метод Task.Run() позволяет создать и запустить задачу для выполнения операции. Кроме того, можно создать экземпляр класса Task с использованием конструктора и запустить его с помощью метода Start();

– раздачи позволяют выполнять операции асинхронно, не блокируя основной поток выполнения. Методы класса Task позволяют выполнять асинхронные операции и ожидать их завершения. Например, метод Task.Delay() позволяет задержать выполнение задачи на определенное время без блокировки потока;

– класс Task предоставляет методы и свойства для управления жизненным циклом задачи. Методы Wait() и WaitAll() позволяют ожидать завершения одной или нескольких задач. Свойства IsCompleted, IsCanceled и IsFaulted позволяют проверить состояние задачи;

– задачи могут возвращать результаты своего выполнения. Для получения этих результатов можно использовать свойство Result, которое содержит возвращаемое значение. Также можно использовать методы ContinueWith() или ключевые слова async/await для асинхронной обработки результатов задачи.

Таким образом, класс Task предоставляет мощный инструментарий для работы с асинхронными операциями в языке C#. Он упрощает разработку асинхронного кода, позволяет избежать блокировки основного потока и эффективно управлять выполнением задач.

3.5.4 Описание SQLite

SQLite является компактной и встраиваемой реляционной базой данных,

которая широко применяется в мобильной разработке на платформе .NET MAUI. Вот некоторая информация о работе с SQLite в контексте разработки мобильных приложений на .NET MAUI:

– SQLite предоставляет полный набор функций SQL и поддерживает стандартные операции баз данных, включая создание таблиц, выполнение запросов, индексирование и управление транзакциями. Благодаря тому, что SQLite хранит данные в одном файле, его можно легко встраивать в приложения .NET MAUI;

– NET MAUI можно использовать библиотеку SQLite-net-pcl, которая предоставляет простой и удобный интерфейс для взаимодействия с базой данных SQLite. Для использования библиотеки необходимо добавить пакет NuGet SQLite-net-pcl в проект .NET MAUI;

– после подключения к базе данных SQLite можно создавать таблицы и определять структуру данных с использованием SQL-запросов или с помощью объектно-реляционных отображений (ORM), таких как Entity Framework или SQLite-net-pcl. С помощью SQL-запросов или методов ORM можно выполнять операции создания, чтения, обновления и удаления данных (CRUD) в таблицах;

– SQLite поддерживает многопоточное использование и может обрабатывать несколько соединений к базе данных одновременно. В разработке приложений .NET MAUI можно управлять открытием и закрытием соединений с базой данных в соответствии с требованиями приложения. Также можно использовать транзакции для обеспечения целостности данных и атомарности операций;

– .NET MAUI можно использовать привязку данных (data binding), чтобы связать данные из базы данных SQLite с элементами управления пользовательского интерфейса, такими как ListView, DataGrid или ComboBox. Это позволяет получать данные из базы данных и отображать их в пользовательском интерфейсе, а также обновлять данные и сохранять изменения обратно в

базу данных;

– если в приложении .NET MAUI требуется вносить изменения в структуру базы данных SQLite, например, добавлять новые таблицы или поля, можно использовать механизм миграции базы данных, такой как SQLite-net-migrations. Это обеспечивает возможность обновления существующей базы данных без потери данных.

Таким образом, использование SQLite в разработке мобильных приложений на .NET MAUI предоставляет мощные возможности для работы с базами данных и обеспечивает эффективное управление данными в приложении.

3.5.5 Разработка пользовательского интерфейса

Для разработки пользовательского интерфейса в .NET MAUI был использован XAML. XAML (Extensible Application Markup Language) является декларативным языком разметки, применяемым в .NET MAUI для описания пользовательского интерфейса (UI) в приложениях. Он обеспечивает удобный способ создания графического интерфейса и разделяет структуру и внешний вид элементов интерфейса от логики приложения.

XAML позволяет описывать элементы пользовательского интерфейса, их свойства, стили, шаблоны, анимации и другие аспекты в виде иерархической структуры объектов, подобной XML. Это делает разделение дизайна и логики более понятным и гибким.

Основная идея XAML заключается в том, что разработчик может создавать и настраивать элементы интерфейса с помощью разметки, без необходимости программирования на языке C# или другом языке. XAML-разметка может быть связана с логикой приложения с использованием привязок данных и событий.

Применение XAML в .NET MAUI позволяет разработчикам более эффективно работать с интерфейсом приложения, улучшает читаемость кода и облегчает совместную работу между дизайнерами и разработчиками.

XAML также реализует паттерн MVVM (Model-View-ViewModel) –

паттерн проектирования, широко используемый в разработке пользовательского интерфейса, особенно в технологиях, таких как WPF и Xamarin. Он помогает разделить логику приложения от представления данных и обеспечивает хорошую организацию и поддержку кода.

Основные компоненты паттерна MVVM включают:

– модель (Model). Представляет бизнес-логику и данные приложения. Включает классы и структуры данных, а также операции, связанные с обработкой данных;

– представление (View). Отображает данные модели пользователю. Это пользовательский интерфейс приложения, состоящий из элементов управления, макетов и графических компонентов;

– представитель (ViewModel). Связывает модель и представление. Он предоставляет данные и команды, необходимые для отображения и взаимодействия с моделью. Представитель отвечает за преобразование данных из модели в формат, понятный представлению, и обрабатывает команды пользователя, взаимодействуя с моделью;

Основные принципы MVVM включают:

– каждый компонент (Модель, Представление, Представитель) имеет свою специфическую роль и отвечает только за свои задачи;

– представление и представитель взаимодействуют с помощью двусторонней привязки данных, что позволяет автоматически обновлять данные в обоих направлениях;

– представитель предоставляет команды, которые могут быть вызваны из представления для выполнения определенных действий. Это позволяет обрабатывать пользовательский ввод и взаимодействовать с моделью.

3.6 Работа с программным обеспечением

При запуске приложения, открывается главное окно, которое иллюстрируется на рисунке 19. Главное окно состоит из центрального изображения, под которым расположены 3 кнопки разделов. В правом верхнем углу заголовка

размещена кнопка профиля, в которой находится форма профиля, представленная на рисунке 20. В данной форме расположена информация о пользователе и текущем автомобиле. По этим данных происходит дальнейшее обращение к элементам других форм.

Рисунок 19 – Главный вид приложения

3.6.1 Подсистема «Обслуживание»

В подсистеме "Обслуживание" должен быть реализован список кнопок, каждая из которых открывает форму для определенной детали автомобиля. Каждая деталь сопровождается соответствующим изображением, а также предоставляет дополнительные поля для записи интересующих деталей. Форма, отображающая данную функциональность, представлена на рисунке 21.

Содержимое формы напрямую зависит от выбранного автомобиля и указанного типа двигателя в профиле. Например, при просмотре формы для автомобиля с бензиновым двигателем будут доступны информация о замене масла и масляных фильтров, в то время как для электрического двигателя будут

представлены соответствующие данные по обслуживанию электрических моторов.

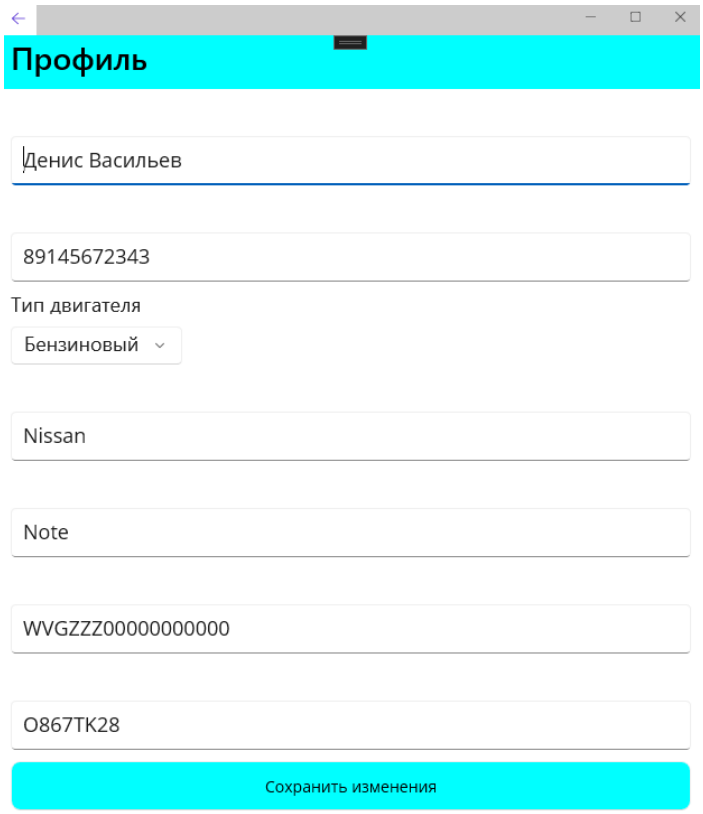

Рисунок 20 – Окно профиля

Кнопки «Личное» содержат пустые формы по запчастям с указание дополнительного поля «Название запчасти». Данные элементы переносятся в базу данных и также учитываются при составлении отчета обслуживания.

В форме "Запчасть" предоставлена информация о конкретной детали автомобиля. При нажатии на кнопку, открывается новая форма, специально предназначенная для данной детали, где пользователю предлагается указать следующую информацию:

– дата замены: указывается дата, когда произошла замена данной детали;

– пробег: указывается пробег автомобиля на момент проведения замены;

– название замененной детали: пользователь вводит название детали, которая была заменена;

– стоимость проведенной замены: указывается стоимость замены данной детали.

Рисунок 21 – Форма «Обслуживание»

Кроме того, при нажатии на кнопку "Личное", пользователь может указать название заменяемого элемента, которое в дальнейшем может быть изменено по его усмотрению. Форма запчасти, где представлена указанная информация, представлена на рисунке 22.

3.6.2 Подсистема «Сервисное обслуживание»

В подсистеме "Сервисное обслуживание" при первом запуске приложения пользователю предлагается отправить запрос на разрешение доступа к его местоположению. При получении разрешения, открывается приложение Google Карты с указанием примерного местоположения пользователя и отображением ближайших сервисных центров. Пример работы этой функции представлен на рисунке 23 для компьютера и на рисунке 24 для мобильного приложения Android.

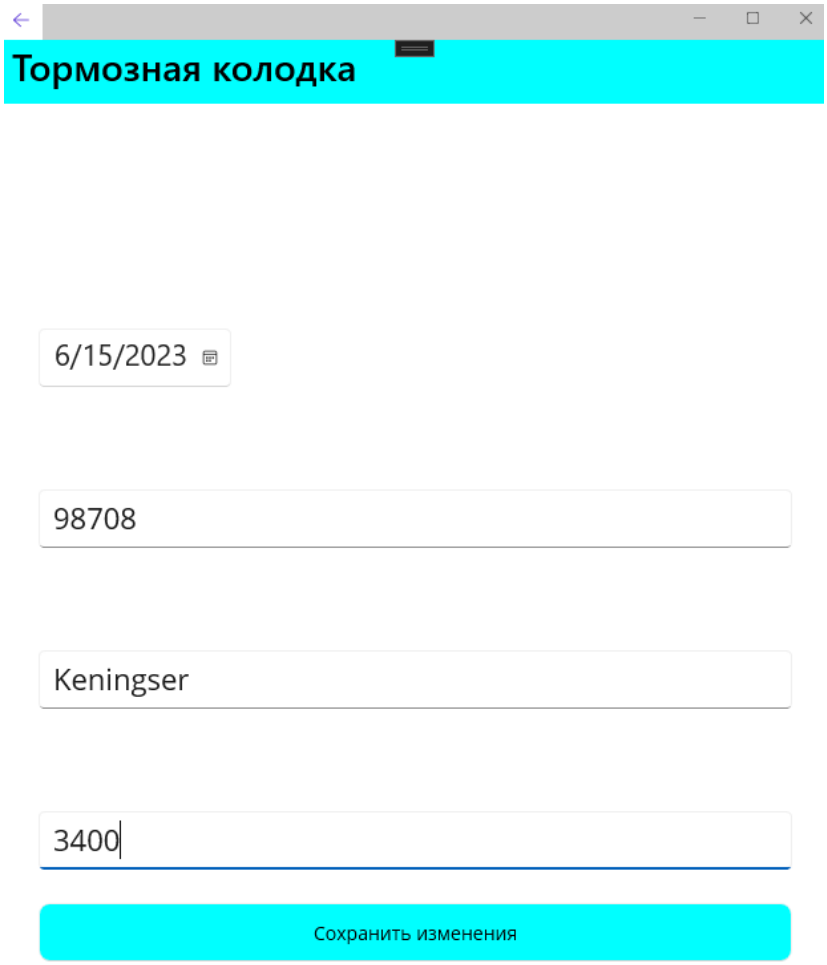

Рисунок 22 – Форма «Запчасть»

Такое взаимодействие с местоположением пользователя позволяет приложению более точно определить его текущее положение и предложить ближайшие сервисные центры для обслуживания автомобиля. Это удобно для пользователей, так как они могут быстро найти ближайший центр и принять информированное решение о выборе места для обслуживания своего автомобиля.

Использование приложения Google Карты для представления этой функциональности гарантирует надежность и удобство, так как Google Карты предоставляют точные географические данные и навигационные возможности. Это позволяет пользователям получить наглядное представление о своем расположении относительно сервисных центров и легко найти наиболее удобный и доступный вариант.

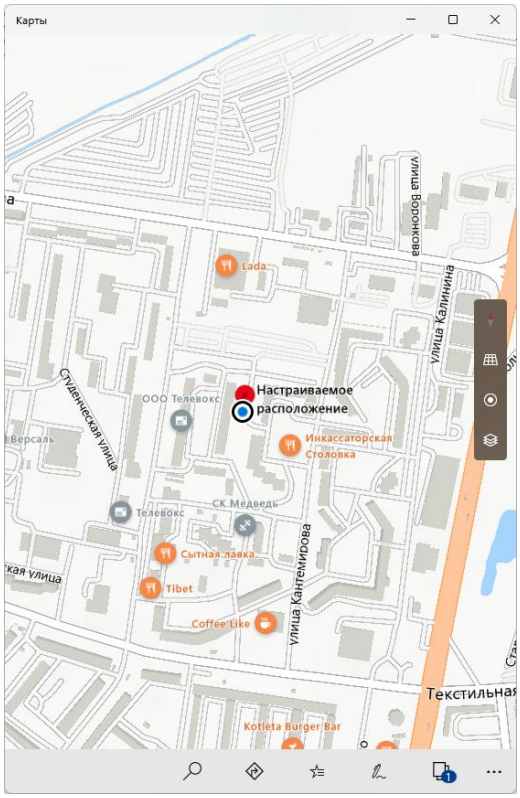

Рисунок 23 – Приложение карт для Windows

Карты для мобильного приложения имеют возможность автоматической авторизации в приложении "Google Карты" и использования текущего профиля пользователя "Google". Это обеспечивает приложению способность сохранять историю поисковых запросов, что полезно для пользователей, позволяя им легко получать доступ к предыдущим поискам и сохраненным местам. В долгосрочной перспективе планируется интегрировать карты непосредственно в приложение, чтобы увеличить удобство использования и сделать его еще более интуитивно понятным для пользователей.

При выборе сервисного центра в приложении будет отображаться полезная контактная информация, включая номер телефона. Такое предоставление контактных данных позволит пользователям легко и удобно связаться с выбранным сервисным центром, например, для уточнения деталей, записи на услугу или задания вопросов. Это создаст более непосредственное и оперативное взаимодействие между клиентами и сервисным центром, улучшая общий опыт обслуживания автомобилей.

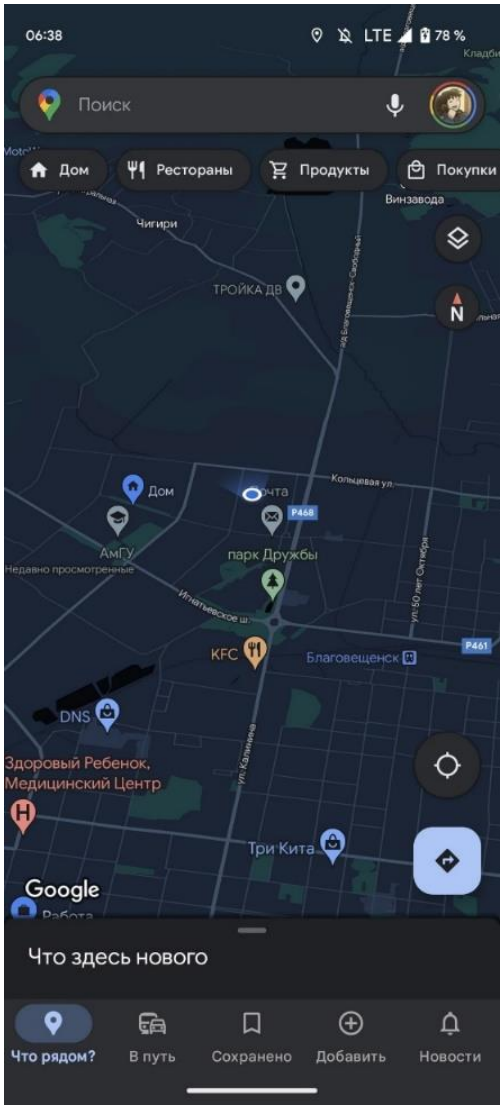

Рисунок 24 – Приложение карт для Android

Таким образом, интеграция карт в мобильное приложение для сервисного обслуживания автомобиля, а также предоставление контактной информации о сервисных центрах, значительно повысит удобство и функциональность приложения. Эти дополнительные функции позволят пользователям более эффективно использовать приложение и сократят время и усилия, связанные с поиском и контактированием с нужным сервисным центром.

3.6.3 Подсистема «QR-код»

Данная подсистема генерирует QR-код, который формируется на основе данных, извлеченных из профиля текущего автомобиля и подсистемы "обслуживание". Затем эта подсистема обращается к соответствующему шаблону

документа в формате dox, постоянно заполняя его актуальными данными о заменах. Такой подход значительно упрощает работу сервисных центров и позволяет предварительно определить, когда необходимо провести замену.

QR-код, сгенерированный подсистемой, содержит в себе информацию о текущем состоянии автомобиля и его обслуживании. При обращении к шаблону документа, соответствующему конкретному автомобилю, подсистема автоматически заполняет данные о предстоящих заменах деталей или выполнении определенных процедур обслуживания. Это позволяет сервисным центрам оперативно ориентироваться в необходимых работах и планировать свою деятельность заранее

Такой подход к управлению данными об обслуживании автомобилей с помощью QR-кодов и шаблонов документов в формате dox обладает несколькими преимуществами. Во-первых, он значительно упрощает процесс сбора и передачи данных о заменах, поскольку информация уже представлена в удобном и структурированном формате. Во-вторых, такой подход обеспечивает актуальность данных, поскольку они автоматически обновляются при каждом изменении в профиле автомобиля или подсистеме "обслуживание". Наконец, подсистема позволяет сервисным центрам предсказывать потребность в замене деталей и планировать свою работу заранее, что способствует повышению эффективности и оптимизации процесса обслуживания автомобилей. Данная форма представлена на рисунке 25.

При сканировании автомобиля сотруднику передается VIN-номер, обеспечивая удобное внесение информации в систему сервисного центра. В результате, для пользователя создается документ в формате Word, содержащий историю технического обслуживания его текущего автомобиля. Эти данные сохраняются в базе данных и остаются доступными на протяжении всего использования программы, обеспечивая надежное хранение информации о прошлых обслуживаниях и ремонтах автомобиля пользователя.

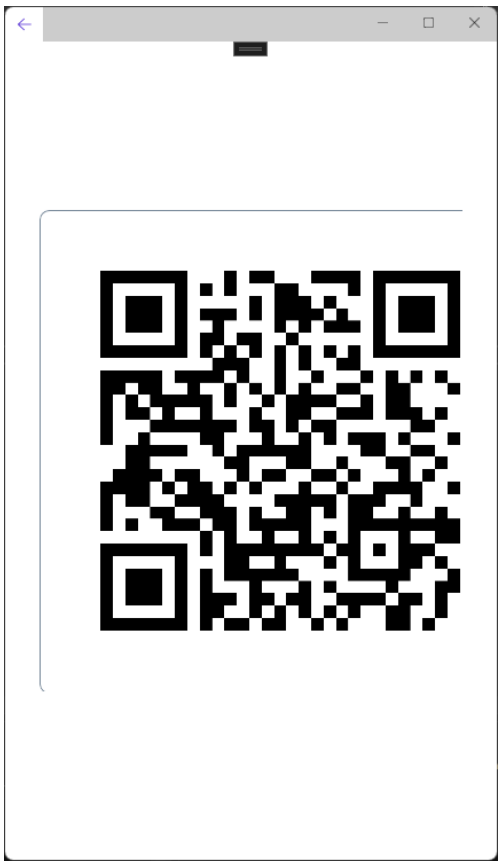

Рисунок 25 – QR-код

Это подход обеспечивает эффективную и надежную систему записи и хранения информации о техническом обслуживании автомобиля. За счет передачи VIN-номера, сотрудники сервисного центра могут быстро и точно идентифицировать автомобиль и оперативно внести все необходимые данные. Пользователь, в свою очередь, получает документ, содержащий подробную историю обслуживания его автомобиля, что полезно для отслеживания проведенных работ и планирования дальнейшего обслуживания.

4 БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ

4.1 Безопасность

В контексте работы с вычислительной техникой, осведомленность оператора о правилах безопасности труда играет важную роль. Знание этих правил позволяет оператору предотвратить возникновение несчастных случаев на производстве и снизить вред, связанный с воздействием на него факторов риска, связанных с трудовой деятельностью. Также они способствуют предотвращению профессиональных заболеваний, которые неотъемлемо сопутствуют рабочему процессу. Одним из ключевых аспектов обеспечения безопасности и здоровья при работе с вычислительной техникой является соблюдение требований, предъявляемых к рабочим местам, оснащенным персональными электронно-вычислительными машинами (ПЭВМ).

Правильное соблюдение требований безопасности на рабочем месте с ПЭВМ является необходимым условием для снижения риска возникновения несчастных случаев, предотвращения травм и обеспечения комфортных условий для эффективной работы. Это включает соблюдение правил эргономики, настройку рабочего места с учетом особенностей оператора и требований, а также регулярное проведение обслуживания и проверки оборудования. Такой подход способствует поддержанию здоровья оператора, повышению его производительности и снижению возможных негативных последствий работы с ПЭВМ.

4.1.1 Требования к помещению для работы с ПЭВМ

Организация помещений для работы с персональными электронно-вычислительными машинами (ПЭВМ) играет важную роль в минимизации негативных последствий, как для пользователей, так и для окружающей среды. Для достижения оптимальных условий функционирования и снижения рисков возникновения нештатных ситуаций и поломок, были разработаны следующие требования к помещениям, в которых размещается вычислительная техника:

Окна в помещении, используемом для эксплуатации ПЭВМ, должны быть ориентированы на север или северо-восток, если они служат источником естественного освещения в дневное время.

Уровень освещенности, обеспечиваемый как естественным, так и искусственным освещением, должен соответствовать нормам, установленным в соответствующих нормативных документах.

Если в помещении, оборудованном вычислительной техникой, имеются оконные проемы, то они должны быть оборудованы специальными устройствами, позволяющими регулировать степень прохождения естественного освещения, такими как жалюзи, занавески, внешние козырьки и другие.

Помещение, где происходит эксплуатация ПЭВМ, должно обладать как искусственным, так и естественным освещением. В случае невозможности обеспечить естественное освещение, его использование допускается лишь при наличии достаточного обоснования и положительного заключения от санитарно-эпидемиологических служб.

Если помещение оборудовано экраном с использованием жидкокристаллических или плазменных технологий, минимальная площадь такого помещения должна составлять не менее 4,5 квадратных метров.

Рабочие места в помещениях, где эксплуатируются ПЭВМ, должны быть оборудованы средствами защитного заземления согласно техническим требованиям, чтобы предотвратить повреждения или возникновение чрезвычайных ситуаций, таких как пожары, вызванные коротким замыканием.

В случае наличия рядом силовых кабелей, высоковольтных линий или трансформаторов, а также технологического оборудования, способного вызывать помехи в работе вычислительной техники, необходимо переместить рабочие места как можно дальше от указанных элементов, чтобы избежать ошибок как программного, так и аппаратного характера при работе с ПЭВМ.

Реализация данных требований обеспечивает оптимальные условия для работы с ПЭВМ, снижает возможные риски и обеспечивает безопасность и

надежность функционирования вычислительной техники в различных областях применения.

4.1.2 Требования к уровням шума и вибрации

Воздействие повышенного уровня шума на работника, выполняющего задачи на вычислительной машине, оказывает негативное воздействие на функционирование сердечно-сосудистой и нервной систем, пищеварительных и кроветворных органов, а также увеличивает риск развития тугоухости, что может привести к полной потере слуха. Превышение установленных нормативов шума также может вызывать нервное истощение и психологическую депрессию. Работа в условиях повышенного шума представляет серьезную угрозу для здоровья работника и значительно снижает его производительность. С целью снижения риска возникновения заболеваний, связанных с высоким уровнем шума, были установлены следующие нормы:

– в помещениях, где осуществляется производственная деятельность и выполняются работы на вычислительной машине, уровень шума должен соответствовать действующим санитарно-эпидемиологическим нормативам для конкретных видов работ;

– в помещениях, где осуществляется эксплуатация персональных электронно-вычислительных машин и проводятся производственные работы, уровень вибрации должен находиться в пределах установленных нормативов;

– если на производстве используется оборудование, создающее шумовую нагрузку, превышающую установленные нормативы, такое оборудование должно быть размещено вне помещения, где находится персональная электронно-вычислительная машина;

– при использовании оборудования, превышающего установленные нормативы по уровню шума, необходимо применять средства и методы, направленные на снижение шума или препятствующие его распространению. Возможным решением является замена шумного оборудования на аналогичные модели с более низким уровнем шума.

4.1.3 Требования к организации рабочих мест с ПЭВМ

Организация рабочего пространства и его влияние на здоровье пользователей вычислительной техники представляют значительную проблему [1]. Неправильная организация рабочего пространства может привести к нарушениям опорно-двигательного аппарата, возникновению сколиоза, нарушениям обмена веществ и образованию тромбов, особенно в нижних конечностях, что может увеличить риск инсульта. Для предотвращения подобных проблем существуют рекомендации по расположению пользователя за компьютером, которые представлены на рисунке 26.

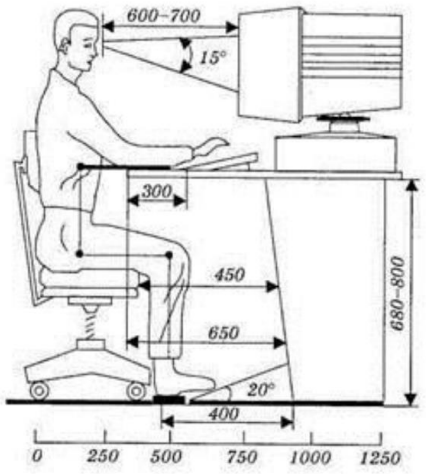

Рисунок 26 – Рекомендуемое размещение пользователя за ПЭВМ

В данной области существуют определенные рекомендации, направленные на организацию рабочего места пользователя вычислительной техники [2]:

При использовании стола необходимо обеспечить наличие регулирующих механизмов, позволяющих регулировать его высоту. Рекомендуемая высота рабочего стола составляет от 680 до 800 миллиметров. В случае отсутствия регулирующего механизма, рекомендуемая высота составляет 725 миллиметров.

При высоте рабочего стола 725 миллиметров, модульные размеры могут быть следующими: ширина – 800, 1000, 1200 и 1400 миллиметров, глубина –

800 и 1000 миллиметров.

Для комфортной работы за столом необходимо обеспечить достаточное пространство для ног пользователя. Высота такого пространства должна быть не менее 600 миллиметров, а ширина – не менее 500 миллиметров. Глубина пространства измеряется на уровне колен и на уровне вытянутых ног. Первоначальная глубина должна быть не менее 450 миллиметров, а вторая – не менее 650 миллиметров.

Создание комфортных условий для работы за столом. Важно обеспечить пользователю возможность регулировать не только высоту стола, но и стула. Рекомендуется использовать сиденье с шириной и глубиной не менее 400 миллиметров, с закругленным передним краем. Сиденье должно быть регулируемым по высоте в пределах от 400 до 550 миллиметров, а также иметь возможность регулировки угла наклона вперед до 15 градусов и назад до 5 градусов. Важно, чтобы точка соприкосновения спины пользователя и спинки стула находилась на высоте 300 миллиметров от сиденья, с возможностью регулировки в пределах 20 миллиметров. Угол наклона спинки стула должен быть регулируемым в пределах 30 градусов. Расстояние от заднего края сиденья до спинки стула должно находиться в пределах от 260 до 400 миллиметров. Длина подлокотников, которые могут быть как стационарными, так и съемными, должна быть не менее 250 миллиметров, а их ширина должна лежать в пределах от 50 до 70 миллиметров.

Обеспечение удобства для ног пользователя. Важно предоставить подставку для ног шириной не менее 300 миллиметров и глубиной не менее 400 миллиметров. Подставка должна быть регулируемой по высоте до 150 миллиметров, а угол наклона опорной поверхности должен регулироваться до 20 градусов.

Расположение клавиатуры. Клавиатура вычислительной машины должна быть размещена на специальной регулируемой поверхности, отделенной от основной столешницы. В случае отсутствия такой поверхности,

клавиатура должна располагаться на расстоянии от 100 до 300 миллиметров от края столешницы.

4.1.4 Микроклимат

Микроклимат производственных помещений представляет собой набор нормированных параметров, таких как температура, влажность, тепловое излучение и другие, которые оказывают влияние на теплообмен человека и влияют на его самочувствие, работоспособность, здоровье и производительность труда. Главная задача охраны труда заключается в поддержании микроклимата рабочего места в пределах гигиенических норм.

На рабочих местах значительное влияние на микроклимат оказывает персональная электронно-вычислительная техника (ПЭВМ), которая повышает температуру человека и приводит к снижению работоспособности и производительности. Кроме того, ПЭВМ также повышает температуру всего помещения в целом. Поддержание температуры на необходимом уровне позволяет обеспечить безопасность и комфортность работы с ПЭВМ.

Для поддержания микроклимата в помещении используются системы вентиляции, которые обеспечивают смену воздуха и подачу чистого воздуха извне. Чтобы создать наиболее комфортные условия, часто применяется система естественной вентиляции. Весной и летом также может использоваться система кондиционирования, чтобы полностью нормировать микроклиматические параметры в рабочем помещении и создать комфортные условия труда.

Системы кондиционирования также используются для поддержания постоянной температуры, влажности и очистки от вредных веществ. Эти системы решают проблему задержки углекислого газа в помещении.

Система отопления обеспечивает заданную, постоянную и равномерную температуру воздуха в рабочих помещениях в холодный период года. Расчет системы отопления осуществляется для компенсации потерь тепла через ограждающие конструкции здания и нагрева холодного воздуха, проникающего в помещение. Существуют различные виды систем отопления, включая

водяные, паровые, воздушные и комбинированные.

Системы водяного отопления являются наиболее эффективными с точки зрения санитарно-гигиенических норм. Они также надежны и позволяют регулировать температуру в широком диапазоне. Такие системы часто используются в помещениях, где находятся рабочие места с ПЭВМ. В холодный период года температура в помещении не должна превышать (22-24) °С, а в теплый период года она должна быть в пределах (20-25) °С. Относительная влажность воздуха должна составлять (40-60) %, а скорость движения воздуха не должна превышать 0,1 м/с.

4.1.5 Требования к графическому интерфейсу

При разработке системы были использованы компоненты, описанные в ГОСТ Р ИСО 9241-161-2016. С графическими элементами интерфейса пользователя можно работать посредством различных способов ввода, использующих различные приемы:

– ввод данных с клавиатуры;

– указание с помощью компьютерной мыши, ручки, распознавания жестов, отслеживания положения глазного яблока;

– речевой ввод с использованием голосовых команд, распознавания голоса.

Если в интерактивной системе используют несколько способов ввода, у пользователя должна быть возможность применения всех таких способов. В данной работе могут использоваться: классический ввод с использованием клавиатуры и мыши, а также ввод без использования отдельного указательного устройства, поскольку использует прямое управление указателем.

Оконное приложение разработано по классическим принципам разработки пользовательского интерфейса. Были использованные элементы и принципы взаимодействия с ними, описанные в ГОСТ Р ИСО 9241-161-2016. Все способы взаимодействия с системой соответствуют современным принципам пользовательского интерфейса, в основном используются классические и

общедоступные средства ввода.

4.2 Экологичность

Активное производство электроники оказывает серьезное отрицательное воздействие на окружающую среду и здоровье людей, несмотря на его положительное влияние на повседневную жизнь. Процессы производства электроники включают химические процессы, которые приводят к образованию и накоплению опасных отходов, содержащих вредные вещества. Некоторые из таких веществ, такие как ртуть, щелочи, никель, цинк и поливинилхлорид, используются в составе электроники, например, в подсветке мониторов и источниках питания.

Использование этих веществ в производстве персональных электронно-вычислительных машин (ПЭВМ) требует специальных методов утилизации, чтобы минимизировать негативное воздействие на окружающую среду. Процесс утилизации вычислительной техники включает следующие шаги. Вначале на предприятии формируется специальная комиссия, которая отбирает технику, подлежащую утилизации. Перед самой утилизацией проводится экспертная оценка, чтобы подтвердить необходимость утилизации выбранной техники. Экспертом может выступать как сотрудник предприятия, так и независимый специалист с соответствующей компетенцией.

Далее составляется акт списания, который отражает причины и количество техники, подлежащей списанию, а также отражается в бухгалтерском учете предприятия. Утилизация техники производится на специализированном предприятии, обладающем разрешением на переработку вычислительной техники. По завершении утилизации, предприятие предоставляет официальный документ, подтверждающий соответствие утилизации требуемым технологиям и отсутствие загрязнения окружающей среды отходами техники [3].

4.3 Чрезвычайные ситуации и методы их предотвращения

Безопасность при эксплуатации персональных электронно-вычислительных машин (ПЭВМ) является важным аспектом, поскольку помещения,

где они используются, подвержены возможным пожарным рискам. Пожары могут возникать из-за короткого замыкания между компонентами ПЭВМ или перегрева ее элементов. В случае возникновения пожара, необходимо соблюдать определенные действия для минимизации ущерба и обеспечения безопасности всех присутствующих.

Первоначально следует немедленно оповестить пожарную службу, сообщив адрес предприятия, место возникновения пожара, свое имя, должность и контактный телефон. Затем необходимо активировать пожарную сигнализацию для оповещения всех присутствующих в здании. План эвакуации должен быть аккуратно выполнен, сбор на улице и проверка наличия всех сотрудников. Если руководитель отсутствует, его следует немедленно уведомить о возникновении пожара. Также необходимо подготовиться к прибытию пожарных подразделений и, при возможности, начать тушение пожара с помощью первичных средств, доступных в организации [4].

При наличии пострадавших следует оказать им первую медицинскую помощь. При термических ожогах необходимо охладить пораженные участки тела. Нельзя промывать открытые раны, смазывать ожоги маслом, вскрывать пузыри или снимать поврежденную одежду. Важно оказать помощь пострадавшим и обеспечить их комфорт.

Для предотвращения возникновения пожароопасных ситуаций рекомендуется принимать профилактические меры. Вентиляция помещения, где находится ПЭВМ, должна обеспечивать поступление свежего воздуха для охлаждения компонентов вычислительной системы. Систематические сервисные работы, включая очистку ПЭВМ от пыли и проверку состояния ее элементов, помогут предотвратить их перегрев и заранее заменить поврежденные компоненты. Горючие предметы и емкости с легковоспламеняющимися жидкостями следует удалить из окружающей среды ПЭВМ. Вентиляционные отверстия в помещениях с ПЭВМ необходимо поддерживать свободными от преград. Рядом с ПЭВМ рекомендуется разместить огнетушитель типа.

ЗАКЛЮЧЕНИЕ

В ходе проведения данного исследования и разработки решения для сервисного обслуживания автомобилей при помощи мобильного приложения были выполнены поставленные задачи. Была определена проблема отсутствия эффективного механизма инвентаризации, а также разработан программный продукт, который успешно решает данную проблему.

Проектирование системы включало подробную разработку архитектуры системы и выделение необходимых модулей для последующей реализации. Для достижения этой цели проведен анализ различных моделей жизненного цикла и архитектуры проекта. Была выбрана оптимальная модель жизненного цикла, которая обеспечила эффективную разработку системы с соблюдением принципов параллельного тестирования и разработки модулей. Это позволило повысить производительность и качество системы, ускорить процесс разработки и обеспечить более гибкое внесение изменений в проект по мере его развития.

Разработка базы данных, соответствующей требованиям заказчика, учитывала полноту и точность данных, необходимых для безошибочного функционирования продукта. Также был разработан план реализации модулей, отвечающих за отображение и взаимодействие данных. Полученные результаты на данном этапе послужили основой для создания схемы взаимодействия моделей системы, что упростило процесс разработки программного продукта.

Разработанное приложение учета технического обслуживания автомобиля предоставляет удобный и простой способ записи данных об автомобиле и отслеживания затрат, связанных с его техническим обслуживанием. Пользователи могут легко внести информацию о своем автомобиле, включая его модель, год выпуска, VIN-номер и другие характеристики. Кроме того, приложение позволяет вносить записи о проведенных технических работах, замене деталей, заправках и других видов обслуживания, а также указывать стоимость

этих услуг.

Одним из главных преимуществ данного программного продукта является его способность упростить ведение отчетности по автомобилям. Пользователи могут легко получить доступ к информации о проведенных работах, затратах, дате их выполнения и других деталях. Это позволяет более эффективно планировать расходы на обслуживание автомобиля и контролировать его техническое состояние. Кроме того, мобильное приложение может предоставлять дополнительные функциональные возможности, такие как напоминания о необходимости периодического технического обслуживания или замены деталей, доступ к истории обслуживания и документации автомобиля, а также возможность обращения к специалистам автосервиса для получения консультаций или помощи в случае аварии.

В итоге, разработка мобильного приложения для сервисного обслуживания автомобиля привносит существенные улучшения в пользовательский опыт, упрощает процесс обслуживания и повышает уровень доверия к автосервисному учреждению. Приложение обеспечивает удобство, эффективность и гарантирует качество предоставляемых услуг, что делает его важной составляющей современной автомобильной индустрии.

БИБЛИОГРАФИЧЕСКИЕ ССЫЛКИ

1 Булгаков А.Б. Безопасность жизнедеятельности [Электронный ресурс]: сб. учеб.-метод. материалов для всех направлений подготовки бакалавров и специалистов / АмГУ, ИФФ; сост. А.Б. Булгаков, В.Н. Аверьянов, М. В. Гриценко. – Благовещенск: Изд-во Амур. гос. ун-та, 2017. http://irbis.amursu.ru/DigitalLibrary/AmurSU_Edition/9036.pdf

2 Кардаш, Т. А. Эргономика рабочих мест служащих и инженерно-технических работников, оснащенных ПЭВМ [Текст]: учеб. пособие / Т. А. Кардаш; АмГУ, ИФФ. - Благовещенск: Изд-во Амур. гос. ун-та, 2002. - 60 с.

3 Безопасность жизнедеятельности в химической промышленности [Электронный ресурс]: учебник / Н. И. Акинин [и др.]. – Санкт-Петербург: Лань, 2019. – Режим доступа: https://e.lanbook.com/book/116363.

4 СанПиН 1.13130.2020 Система противопожарной защиты. Эвакуационные пути и выходы. Введ. 2020-09-19

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Безруков А. И. — Разработка мобильных приложений для платформы Android. / А.И. Безруков // БХВ-Петербург, – 2017.

2 Власов В. В. — Мобильные приложения на платформе Android: создание и разработка. / В.В. Власов // БХВ-Петербург, – 2018.

3 Грунин А. В. — Разработка мобильных приложений на платформе iOS. / А.В. Грунин // ДМК Пресс, – 2016.

4 Инновации и научно-техническое творчество молодежи. 20-21 апреля 2022 г: материалы конференции / RU. – Новосибирск: СибГУТИ, 2022. – 1323 с. – ISBN 978-5-91434-072-5. – Текст: электронный // Лань: электронно-библиотечная система. – URL: https://e.lanbook.com/book/257177 (дата обращения: 07.05.2023).

5 Калачев А. А. — Программирование мобильных устройств на Android. / А.А. Калачев // Лань, – 2017.

6 Колесников Г. В., Рожнова Н. В., Шаляпина О. А. — Разработка мобильных приложений для платформы iOS. / Г.В. Колесникова [и др.] // БХВ-Петербург, – 2019.

7 Курбатов С. В. — Разработка мобильных приложений на базе платформы .NET MAUI. / С.В. Курбатов // Питер, – 2017.

8 Павленко А. В., Матвеева О. В. — Мобильная разработка на платформе .NET MAUI. / А.В. Павленко [и др.] // БХВ-Петербург, – 2018.

9 Просекин А. А. — Разработка приложений для iOS на языке C#. / А.А. Просекин // БХВ-Петербург, – 2018.

10 Сафонов С. Н. — Разработка Android-приложений для профессионалов. / С.Н. Сафонов // ДМК Пресс, – 2017.

11 Хатыпов А. Ш. — Разработка мобильных приложений на базе платформы Android. / А.Ш. Хатыпов // БХВ-Петербург, – 2018.

ПРИЛОЖЕНИЕ А