**ВВЕДЕНИЕ**

Автомобильный парк в Республике Беларусь, как и во всём мире в целом, увеличивается и развивается с высокой скоростью. И, конечно, у этого есть свои преимущества и недостатки. К преимуществам можно отнести: универсальность, высокую доступность, возможность использования различных маршрутов и схем доставки и многое другое. Недостатками считаются: низкая экологичность и безопасность движения, большая энергоёмкость и трудоёмкость, высокая себестоимость перевозок. Разумеется, это касается всего транспорта – и общественного, в частности.

Общественный транспорт – существенная часть современной жизни людей. И поэтому ему необходимо соответствовать инфраструктуре города и удовлетворять всем требованиям населения. Очевидно, что с развитием вычислительной техники и программных средств появляются возможности разработки комплексных технологий для усовершенствования и автоматизации управления общественным транспортом. В настоящее время накоплен большой опыт применения интеллектуальных методов для различных прикладных задач.

Одной из таких проблем является оптимизация маршрутов общественного транспорта, так как некачественно сформированная маршрутная сеть влечёт за собой ухудшение эффективности муниципального транспортно-пассажирского сообщения, повышение затрат на осуществление перевозок и соответственно уменьшение прибыли.

Одна из задач оптимизации основывается на уменьшении числа поездок маршрутных такси или иного общественного транспорта с числом пассажиров меньшим, чем то, на которое рассчитано транспортное средство. Организация движения транспортного средства по определенному маршруту может быть достаточно затратной, поскольку необходимо выделить само транспортное средство, обеспечить его топливом, содержать транспортное средство в исправном состоянии, периодически осуществляя его ремонт и, конечно же, выделить человека на роль водителя.

Все это ставит вопрос об эффективности движения транспортных средств по определенным маршрутом, на которых случаются ситуации, когда реальное число пассажиров значительно ниже того, на которое рассчитано транспортное средство или наоборот выше и не может быть обработано в полной мере.

Целью данного дипломного проекта является разработка мобильного приложения как части системы, позволяющей отслеживать точное число пассажиров на каждый конкретный маршрут в конкретный момент времени, предоставлять более подходящее транспортное средство, основываясь на количестве пассажиров, а также в целом оптимизировать число задействованной техники, сохраняя при этом приемлемый уровень предоставляемых услуг.

Также данная система позволяет собирать статистику на каждый из маршрутов. Подобная информация позволяет понять, насколько эффективен данный маршрут, что в нём можно изменить и т.д. Это позволит проектировать новые маршруты и рационально планировать работу подвижного состава в зависимости от загруженности тех или иных остановок и линий. На основе собранной статистики станет возможным создание нейронной сети, которая автоматически распределяла бы транспорт по маршрутам, выполняла прогнозирование загруженности конкретного маршрута в определенный момент времени.

1. **СИСТЕМНЫЙ АНАЛИЗ И ПОСТАНОВКА ЗАДАЧИ**
   1. **Описание объекта автоматизации**

Ни одно государство в мире в своем историческом развитии не обходилось и не обойдется без развитой транспортной инфраструктуры. В жизни современного города важной составной частью является пассажирский транспорт, основной задачей которого является обеспечение потребности населения в перевозках при систематическом улучшении качества обслуживания пассажиров. Транспортная подвижность жителей и средняя дальность их поездок растет по мере роста численности и городской территории. В соответствии с этим дальнейшее развитие, совершенствование и улучшение качества обслуживания пассажирских перевозок актуально для изучения и реализации.

Одной из важнейших проблем пассажирских перевозок на сегодняшний день является низкий уровень их организации. Это связано с неупорядоченным планированием маршрутов движения пассажирских автотранспортных средств. В связи с повышением уровня автомобилизации и увеличением подвижности населения на фоне недостаточных темпов развития дорожной сети остро стоит проблема оптимизации пассажирских перевозок, направленная на динамическую адаптацию их к постоянно меняющимся условиям.

Примером такого общественного транспорта может служитьмаршрутное такси — микроавтобусы, осуществляющие перевозку пассажиров и багажа по установленным маршрутам и не полностью интегрированные в регулярную систему общественного транспорта. Маршрутное такси не муниципальный вид общественного транспорта, который может иметь одного оператора, обслуживающего сразу несколько маршрутов. Отличительной чертой маршрутного такси является возможность как дублировать муниципальные (регулярные) маршруты, так и иметь собственные тарифы.

Главные недостатки маршрутного транспорта связаны со значительными эксплуатационными затратами, небольшой вместимостью транспортных средств, загрязнением окружающей среды, высоким уровнем шума, непостоянным графиком работы. Но благодаря преимуществам маршрутного транспорта перед другими видами и, несмотря на присущие ему недостатки, оно получило значительное распространение.

В данный момент организации транспорта городов Беларуси не имеют достоверной информации о перевозимых пассажирах по часам суток. Это вызывает определенные трудности в планировании графика движения, так как неизвестно, сколько может понадобиться маршруток на перевозку.

Одно из возможных решений – создать систему, позволяющую пользователям регистрировать в системе заявки на проезд маршрутным транспортом. Это позволит рассчитывать загруженность маршрута на ближайшее время и, как следствие, корректировать количество транспортных средств на конкретном маршруте.

Рассмотрим понятие пассажиропотока. Он характеризуется:

* мощностью или напряжённостью, то есть количеством пассажиров, которое проезжает в определённое время на заданном участке маршрута в одном направлении;
* объёмом перевозок пассажиров, то есть количеством пассажиров, перевозимых автобусами за определённый промежуток времени (час, сутки, месяц, год);
* пассажирооборотом, то есть транспортные работы, выполняемые при перевозке пассажиров.
  1. **Цели и причины разработки системы**

Основной задачей данного дипломного проекта является создание мобильного приложения для системы регистраций заявок на перевозку пассажиров городским общественным транспортом.

Обуславливается эта задача тем, что на текущем этапе развития системы городского общественного транспорта существует такая проблема как движение маршрутных такси по маршрутам, на которых число реальных пассажиров значительно ниже, чем количество мест, доступное в транспортном средстве. Потенциальные клиенты не имеют возможности получить нужную им информацию об актуальном графике движения маршрутного такси и предпочитают другие виды транспорта. Это приводит к необоснованным тратам на топливо и потере выручки для операторов.

Целью разрабатываемой системы является точное фиксирование числа пассажиров на каждом конкретном маршруте и, основываясь на собранной статистике, оптимизация движения транспортных средств на этих самых маршрутах.

* 1. **Обследование объекта автоматизации**

В ходе проведенного обследования объекта автоматизации были получены следующие выводы: для эффективного планирования и управления маршрутами

пассажирского общественного транспорта, числа активных пассажирских транспортных средств, контроля за загруженностью транспортных маршрутов необходимы программные инструменты, позволяющие моделировать движение городских пассажирских транспортных средств и пассажиропоток, а также для прогнозирования результатов изменений, позволяющие упростить работу путём автоматизации расчётов с применением методов математической статистики.

При анализе экономических факторов и проблем объекта автоматизации можно заметить, что улучшение по одному из показателей (например, уменьшение времени ожидания пассажира) может привести к ухудшению другого показателя (к уменьшению заполняемости транспортного средства).

Система заявок, статистика по движению транспортных средств и наполняемости пассажирами позволит эффективнее контролировать необходимое количество активных единиц транспорта для получения оптимальной прибыли или уменьшения расходов на избыточные единицы транспорта.

* 1. **Определение требований к системе**

В ходе выполнения дипломного проекта разрабатывается мобильное приложение, которое позволяет решить ряд проблем, выявленных в ходе проведения обследования.

Приложение как часть системы предназначено для автоматизации следующих процессов транспортных предприятий города:

* + - изучение пассажиропотоков;
    - формирование статистики.

Основные задачи:

* + - определение остановки, на которой находится пользователь посредством QR-кода;
    - предоставление полной информации по маршрутам, проходящим через данную остановку;
    - формирование заявок на перевозку и отправка их на сервер системы.

Целью данного дипломного проекта является создание максимально простого для понимания и использования приложения с графическим интерфейсом для осуществления взаимодействия пользователей с системой транспортных перевозок.

**Общие требования к продукту.**

Данный продукт может быть использован любым человеком, обладающим

минимальными знаниями о работе с приложениями мобильного устройства.

Мобильное приложение запускается на устройстве пользователя. Необходимо предоставить права на использование камеры и соединения по сети интернет для осуществления обмена данными с сервером системы.

Новый функционал может быть добавлен в приложение со следующими версиями продукта.

**Нефункциональные требования.**

Принципы работы приложения должны быть максимально простыми для понимания, интерфейс должен быть удобными в использовании. Максимально низкий порог входа для использования приложения.

**Функциональные требования.**

Функциональные требования к приложению сводятся к следующим пунктам:

* + - возможность распознавания QR-кодов;
    - взаимодействие с базой данных и сервером системы;
    - отображение данных об остановках и маршрутах пользователю;
    - обработка ошибок.

1. **ПРОЕКТИРОВАНИЕ КЛИЕНТСКОЙ КОМПОНЕНТЫ**
   1. **Выбор среды программирования**

С учетом требований и первоначального анализа, был определен список используемых инструментов для реализации программной части:

* основные средства разработки: Java;
* среда разработки: Android Studio;
* технологи: REST API, JSON.

Далее будет приведено описание используемых инструментов.

**Java** — строго типизированный объектно-ориентированный язык программирования общего назначения. Язык Java активно используется для создания мобильных приложений под операционную систему Android. При этом программы компилируются в нестандартный байт-код для использования их виртуальной машиной Dalvik (начиная с Android 5.0 Lollipop виртуальная машина заменена на ART). Для такой компиляции используется дополнительный инструмент, а именно Android SDK (Software Development Kit), разработанный компанией Google.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) — программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор. Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы.

Основные возможности Java:

* автоматическое управление памятью;
* расширенные возможности обработки исключительных ситуаций;
* богатый набор средств фильтрации ввода-вывода;
* набор стандартных коллекций: массив, список, стек и т. п.;
* наличие простых средств создания сетевых приложений;
* наличие классов, позволяющих выполнять HTTP-запросы и обрабатывать ответы.

**Android Studio** — интегрированная среда разработки (IDE) для работы с платформой Android. Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, — официальное средство разработки Android приложений. Данная среда разработки доступна для Windows.

В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками. Такая опция позволяет быстрее опубликовать версию мобильного приложения в Google Play App Store. Для этого присутствует также поддержка инструментов LINT, Pro-Guard и App Signing.

Новые функции появляются с каждой новой версией Android Studio. На данный момент доступны следующие функции:

* расширенный редактор макетов: WYSIWYG, способность работать с UI компонентами при помощи Drag-and-Drop, функция предпросмотра макета на нескольких конфигурациях экрана;
* сборка приложений, основанная на Gradle;
* различные виды сборок и генерация нескольких .apk файлов;
* рефакторинг кода;
* статический анализатор кода (Lint), позволяющий находить проблемы производительности, несовместимости версий и другое;
* встроенный ProGuard и утилита для подписывания приложений;
* шаблоны основных макетов и компонентов Android;
* встроенная поддержка Google Cloud Platform, которая включает в себя интеграцию с сервисами Google Cloud Messaging и App Engine;
* Android Studio 2.1 поддерживает Android N Preview SDK, а это значит, что разработчики смогут начать работу по созданию приложения для новой программной платформы;
* новая версия Android Studio 2.1 способна работать с обновленным компилятором Jack, а также получила улучшенную поддержку Java 8 и усовершенствованную функцию Instant Run.

**JSON** — текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми.

Несмотря на происхождение от JavaScript, формат считается независимым от языка и может использоваться практически с любым языком программирования.

JSON-текст представляет собой одну из двух структур:

* набор пар ключ: значение. Ключом может быть только строка, значением — любая форма. Повторяющиеся имена ключей допустимы, но не рекомендуются стандартом; обработка таких ситуаций происходит на усмотрение программного обеспечения, возможные варианты — учитывать

только первый такой ключ, учитывать только последний такой ключ,

генерировать ошибку;

* упорядоченный набор значений. Во многих языках это реализовано как массив, вектор, список или последовательность;

Особенности работы с JSON:

* JSON - это чисто формат данных - он содержит только свойства, без методов;
* JSON требует двойных кавычек, которые будут использоваться вокруг строк и имён свойств (одиночные кавычки недействительны);
* даже одна неуместная запятая или двоеточие могут привести к сбою JSON-файла и не работать;
* JSON может принимать форму любого типа данных, допустимого для включения в JSON, а не только массивов или объектов (например, одна строка или номер будут действительным объектом JSON);
* в отличие от кода JavaScript, в котором свойства объекта могут не заключаться в двойные кавычки, в JSON в качестве свойств могут использоваться только строки, заключённые в двойные кавычки.

**REST** (Representational State Transfer — «передача состояния представления») — архитектурный стиль взаимодействия компонентов распределённого приложения в сети. REST представляет собой согласованный набор ограничений, учитываемых при проектировании распределённой гипермедиа-системы. В определённых случаях это приводит к повышению производительности и упрощению архитектуры. В широком смысле компоненты в REST взаимодействуют наподобие взаимодействия клиентов и серверов во Всемирной паутине. REST является альтернативой RPC.

В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно GET или POST; такой запрос называют «REST-запрос»), а необходимые данные передаются в качестве параметров запроса.

Для веб-служб, построенных с учётом REST (то есть не нарушающих накладываемых им ограничений), применяют термин «RESTful».

Свойства архитектуры, которые зависят от ограничений, наложенных на REST-системы:

* производительность — взаимодействие компонентов системы может являться доминирующим фактором производительности и эффективности сети с точки зрения пользователя;
* масштабируемость для обеспечения большого числа компонентов и взаимодействий компонентов.
  1. **Модульная структура ПО и ее описание**

Задачей, на решение которой направлен данный дипломный проект, является создание мобильного приложения, позволяющего создавать заявки в системе на перевозку пассажиров общественным транспортом. Это позволит снизить число поездок общественного транспорта по маршрутам, где среднее количество пассажиров ниже того, на которое рассчитано транспортное средство, а также становится возможным ведение статистики по каждому из доступных маршрутов, что позволит анализировать и оптимизировать движение транспортных средств. Всё это влечёт за собой снижение экономических расходов на поддержание данной сети перевозок.

**Приложение клиента-пассажира.**

Начальный экран приложения клиента-пассажира содержит весь функционал, необходимый для формирования заявки. Пользователю предоставляется возможность формирования заявок в двух режимах: с выбором желаемого маршрута или с выбором остановок. Так же есть возможность отсканировать QR-код для автоматического выбора начальной остановки. Приложение осуществляет фоновую проверку на наличие прав использовать камеру мобильного устройства.

Макет данного экрана представлен на рисунке 2.1:

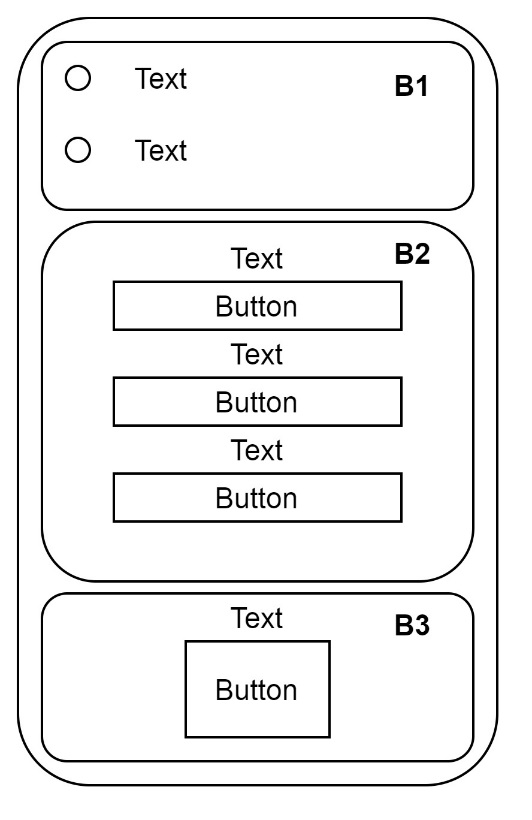


Рисунок 2.1 – Макет начального экрана

Верхняя часть, обозначенная как «B1» (блок 1), содержит две кнопки для выбора режима работы приложения, который больше подходит пользователю.

Центральная часть, обозначенная как «В2» (блок 2), содержит 3 кнопки, с описанием их назначения. Однако на момент начала работы приложения данные кнопки скрыты от пользователя, и он не имеет к ним доступа. После выбора режима работы приложения начнут последовательно появляться кнопки для формирования запроса. Таким образом мы обеспечиваем интуитивно понятный пользовательский интерфейс и избегаем возможных ошибок при формировании запроса.

При нажатии на каждую из трех кнопок происходит одинаковая последовательность действий: отправляется запрос на сервер на получение некоторого списка (номеров маршрутов или же остановок) по средствам мобильного интернета. Клиент получает ответ от сервера и ему открывается новый экран.

Нижний блок, обозначенный как «В3», содержит кнопку для перехода в режим сканирования QR-кода, что может упростить пользователю выбор начальной остановки. Пользователю достаточно просто навести объектив камеры на QR-код и начальная остановка автоматически будет выбрана. Каждый подобный QR-код содержит в себе цифровое значение – уникальный идентификатор остановки, по которому в дальнейшем осуществляется выбор и загрузка данных посредством мобильного интернета.

Система QR-кодов предлагает 4 способа кодировки: цифровое, буквенно- цифровое, байтовое кодирования, а также тип кодирования «кандзи». QR-коды данного приложения используют цифровое кодирование, так как оно является наиболее оптимальным для при решении поставленной задачи. Данный тип кодирования требует 10 бит на 3 символа. Вся последовательность символов разбивается на группы по 3 цифры, и каждая группа (трёхзначное число) переводится в 10-битное двоичное число и добавляется к последовательности бит. Если общее количество символов не кратно 3, то, если в конце остаётся 2 символа, полученное двузначное число кодируется 7 битами, а если 1 символ, то 4 битами.

Например, есть строка «12345678», которую надо закодировать. Последовательность разбивается на числа: 123, 456 и 78, затем каждое число переводится в двоичный вид: 0001111011, 0111001000 и 1001110, и объединяется это в один битовый поток: 000111101101110010001001110.

На момент написания дипломного проекта город Брест, для которого проектируется система, насчитывает всего 515 остановок, что дает возможность формировать уникальные идентификаторы и кодировать их при помощи всего лишь 10 бит. Приложение использует «обычные» QR-коды небольшого размера: 21х21 пиксель (см. рисунок 2.2).

Рисунок 2.2 – Пример QR-кода

Макет экрана, на который приложение переводит пользователя после нажатия на одну из кнопок блока «B2», указанного на рисунке 2.1 представлен на рисунке 2.3:

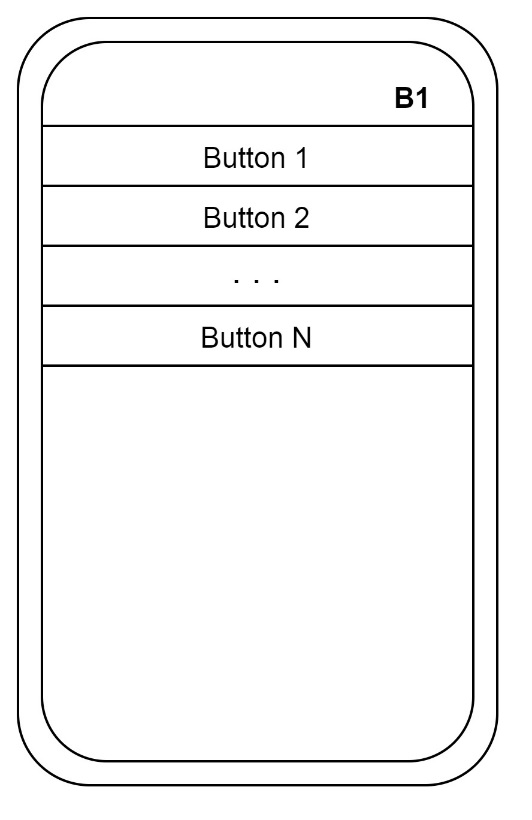
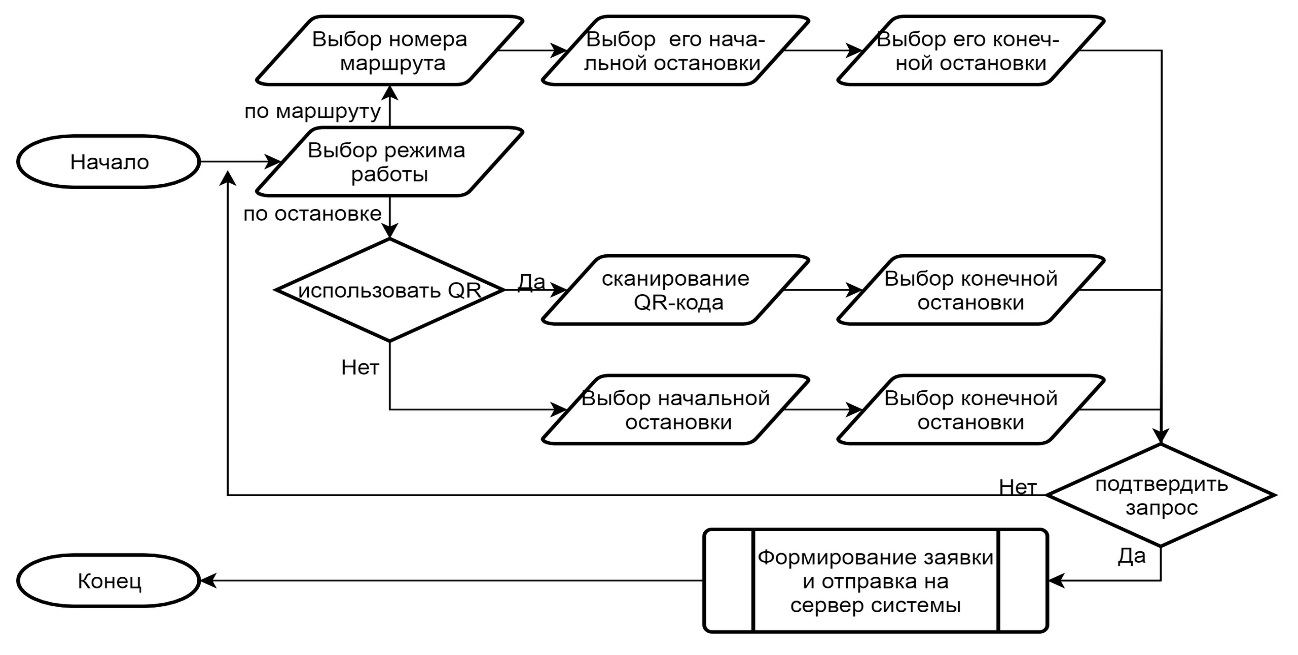


Рисунок 2.3 – Макет окна выбора

На рисунке 2.3 представлен общий макет для окна выбора номера маршрута, начальной и конечной остановок. На данный экран приложения пользователь попадает при нажатии на одну из 3-х кнопок выбора маршрута, находящихся на начальном экране. В зависимости от того, куда нажал пользователь, на сервер отправится определенный запрос и ему будут показаны соответствующие данные. Так, при нажатии на кнопку «выбор маршрута» в новом окне будут отображаться номера активных, на данный момент времени, маршрутов. При выборе кнопки «начальная остановка» в новом окне отобразится список всех доступных остановок или остановок конкретного маршрута (зависит от выбранного режима работы приложения). Так же и с кнопкой «конечная остановка» - предоставляется список

остановок выбранного маршрута или тех, до которых можно добраться от начальной без пересадок, если маршрут не был выбран.

Таким образом работу приложения клиента-пассажира можно обобщить и свести к обобщенной схеме, представленной на рисунке 2.4.

Рисунок 2.4 – Схема работы приложения клиента-пассажира

**Приложение клиента-водителя.**

Начальный экран приложения клиента-водителя предназначен для входа в систему. Содержит 2 поля для ввода: логин и пароль, а также кнопку «Войти». Его макет представлен на рисунке 2.5:

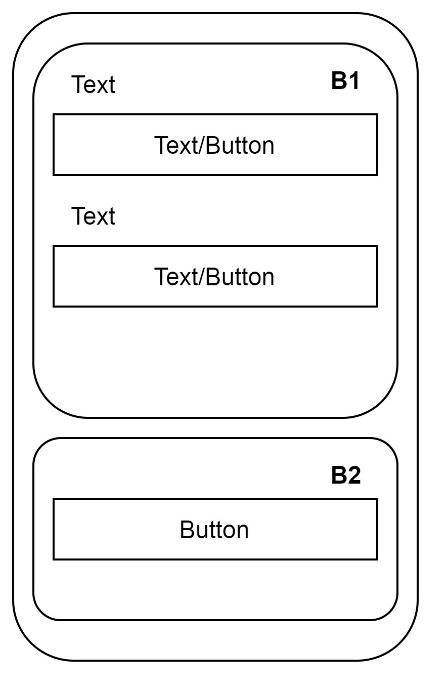


Рисунок 2.5 – Макет начального экрана

Как видно из рисунка 2.5, экран будет состоять из 2 основных частей: блока «В1» и блока «В2». В первом блоке пользователь вводит свои личные данные: логин и пароль (предоставляются заранее). Во втором блоке находит кнопка «Войти», нажав на которую отправляется запрос на сервер и, в случае если данные введены не верно, возвращается ошибка. Иначе пользователю показывает следующее окно, макет которого описан на рисунке 2.6:

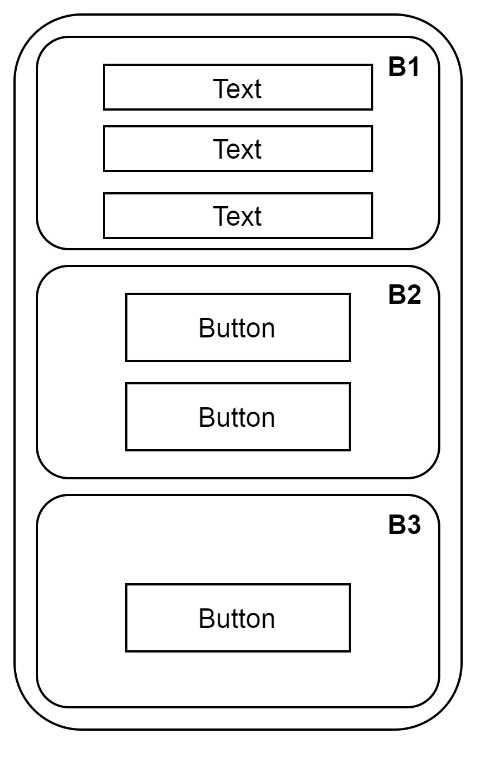


Рисунок 2.6 – Макет основного экрана

На рисунке 2.6 продемонстрирован макет основного, рабочего экрана, который водитель будет видеть на протяжении рабочего дня. Он поделен на 3 основных блока: блок «В1» - информативный, блок «В2» - функциональный, блок «В3» - специальный.

В первом блоке водителю будет динамически отображаться следующая информация: текущий номер маршрута, ближайшая остановка для посадки/высадки пассажиров, количество пассажиров в транспорте на текущий момент. Так как велика вероятность, что не все пассажиры будут использовать приложение для регистрации заявки, то указанное системой количество пассажиров будет не соответствовать реальному положению вещей.

Второй блок («В2») создавался как раз с целью решения проблемы определения фактического количества пассажиров в транспорте. У водителя на экране будут находиться две большие кнопки «+1» и «-1», которыми можно корректировать количество пассажиров. Данный подход позволит не только улучшить качество предоставляемых услуг, т.к. пассажиру будет предоставляться актуальная информация, но и позволит вести статистику соотношения людей,

использующих приложение к не использующим его. Сбор этих данных особенно важен для оценивания динамики роста числа пользователей.

Третий блок («В3») предназначен для внештатных ситуаций и позволяет водителю выйти с аккаунта. Т.к. на стороне сервера используется специальная технология, позволяющая регулировать время нахождения в аккаунте, то кнопка «Выйти из системы» может и не пригодиться. Но в случаях, когда водителю приходится сойти с маршрута (поломка, авария, другие причины), этот блок может быть использован.

Таким образом работу приложения клиента-водителя можно обобщить и свести к обобщенной схеме, представленной на рисунке 2.7:

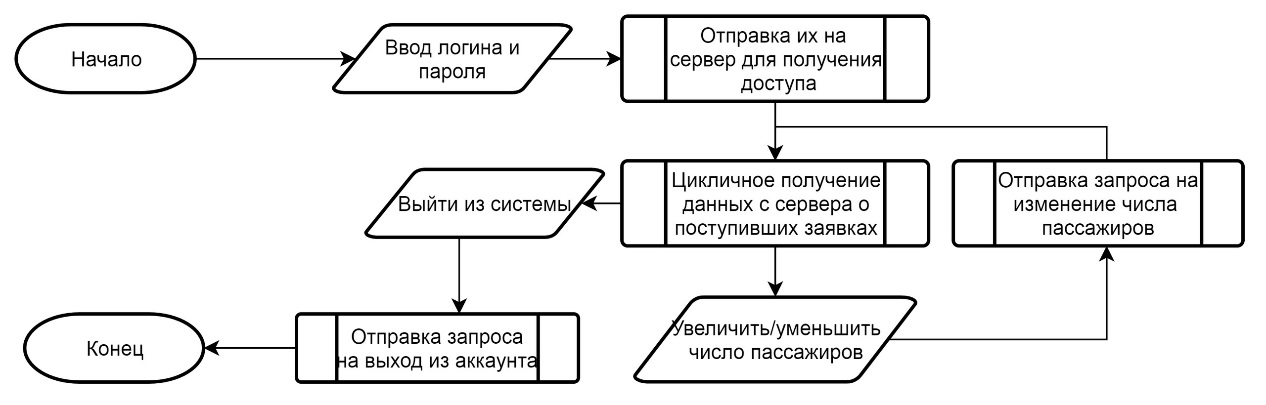


Рисунок 2.7 – Макет основного экрана

1. **РЕЗУЛЬТАТЫ РАЗРАБОТКИ И ИСПЫТАНИЙ СИСТЕМЫ**
   1. **Этапы разработки**

Разработка клиентской части автоматизированной системы управления маршрутным транспортом велась в приложении Android Studio. Данное приложение предоставляет широчайший набор возможностей для разработчика: гибкая настройка среды разработки, выбор различных версий систем (android 5.0, android 9.0 и другие), инструменты для интерактивного построения графической составляющей программы(см. рисунок 3.1), множество библиотек (в том числе для чтения QR-кода), возможность использования строенного эмулятора для тестирования разработанного приложения прямо в Android Studio или динамической отладки приложения через периферийное устройство подключенное через порт USB.

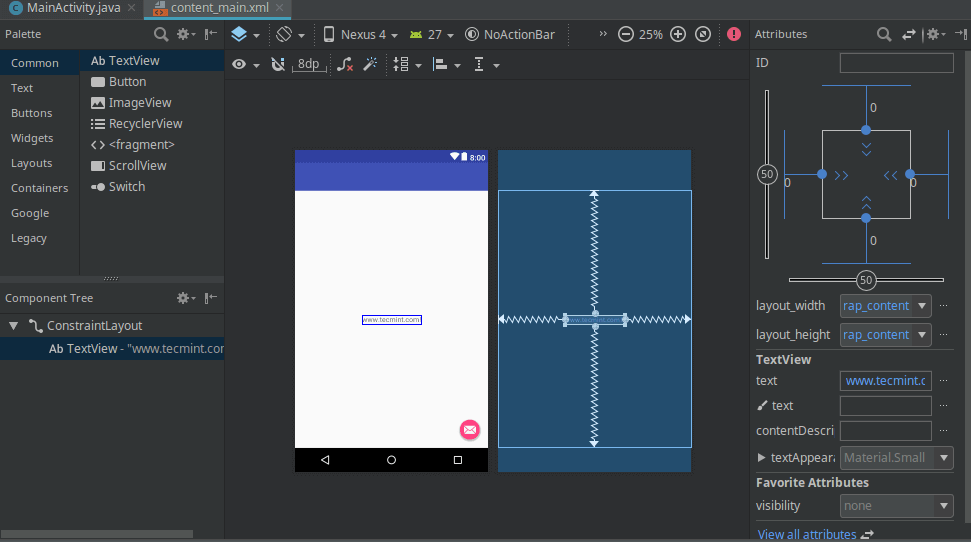


Рисунок 3.1 – графические инструменты Android Studio

Рассмотрим так же и структуру проекта, представленную на рисунке 3.2.

Как видно из рисунка 3.2, проект имеет несколько разделов. Так, папка «Activity» содержит процессы, в которых и будет написан основной код. Папка «models» содержит модели представления информации в запросах, а «res» – ресурсы, которые будут использоваться в ходе разработки (заставки, картинка, стили). В том числе есть конфигурационные файлы, файлы манифеста и многие.

В целом разработку каждого из приложений можно поделить на 3 этапа: создание GUI (Graphical User Interface), создание функционала (написание функций, отправка запросов на сервер, алгоритмы работы с данными GPS), тестирование и исправление недочетов.

Графический интерфейс для приложений создавался при помощи редактора (см. рисунок 3.1).

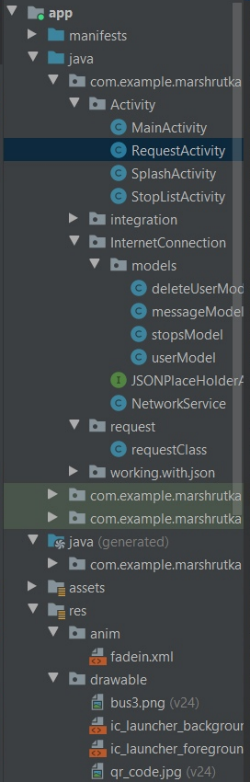


Рисунок 3.2 – структура проекта

Далее проходил процесс создания функционала, который и делает пользовательский интерфейс живым. Данный этап включает в себя написание кода, по обработке действий пользователя. В нашем случае он направлен на отправку запросов серверу для получения какой-либо информации (см. рисунок 3.3), которая будет показана пользователю. А также на недопущение поломки приложения в следствие действий пользователя (отлов багов) путем наложения ряда ограничений или условий для использования.

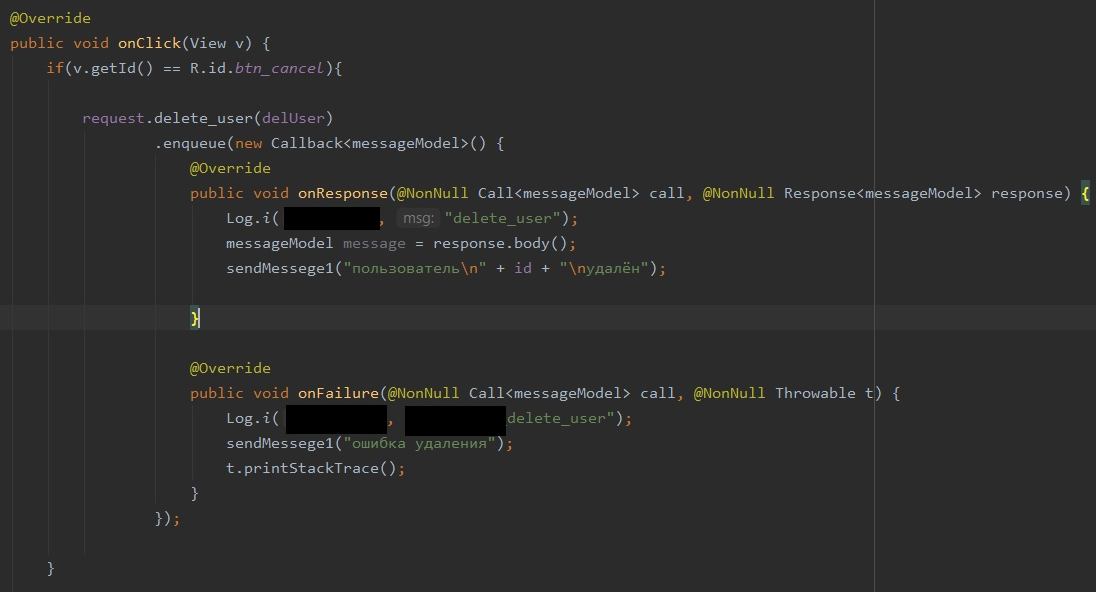


Рисунок 3.3 – пример запроса серверу

* 1. **Цели проведения процесса тестирования**

Тестирование программного продукта представляет собой процесс, направленный на выявление ошибок.

Процесс тестирования включает в себя:

* процесс проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путём наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определённым образом;
* процесс исследования программного обеспечения с целью получения информации о качестве продукта;
* действия, направленные на выявление ошибок;
* диагностику и локализацию ошибок;
* внесение исправлений в программу с целью устранения ошибок.

Большая трудоемкость тестирования и ограниченные ресурсы приводят к необходимости систематизации процесса и методов тестирования.

Цели тестирования могут отличаться, в зависимости от этапа разработки ПО, на котором оно проводится. К примеру, на этапе кодирования целью тестирования будет вызов как можно большего количества сбоев в работе программы, что позволит локализовать и исправить дефекты. В то же время, при приемочном тестировании необходимо показать, что система работает правильно. В период сопровождения, тестирование в основном необходимо для того, чтобы убедится в отсутствии новых багов, появившихся во время внесения изменений.

Тестирование должен проходить любой программный продукт, прикладные программы информационной системы. Помимо обнаружения имеющихся в прикладных программах и, возможно, в структурах базы данных ошибок, сбор статистических данных на стадии тестирования позволяет установить показатели надежности и качества созданного программного обеспечения.

* 1. **Результаты испытаний**

Предварительное тестирование проходило в эмуляторе приложения Android Studio под управлением операционной системы Android 9. Рабочее тестирование проводилось на смартфоне Xiaomi Redmi Note 7 под управлением операционной системы Android 10.

При запуске приложение сначала показывает пользователю загрузочный

экран, на котором изображен логотип приложения (см. рисунок 3.4):



Рисунок 3.4 – Загрузочный экран

После чего пользователь видит главный экран, в котором и будет формироваться запрос (см. рисунок 3.5). Отображение экрана будет меняться в зависимости от выбранного режима работы. Так, выбрав формирование заявки с указанием лишь начальной и конечной остановок пользователю не будет отображаться кнопка выбора маршрута.

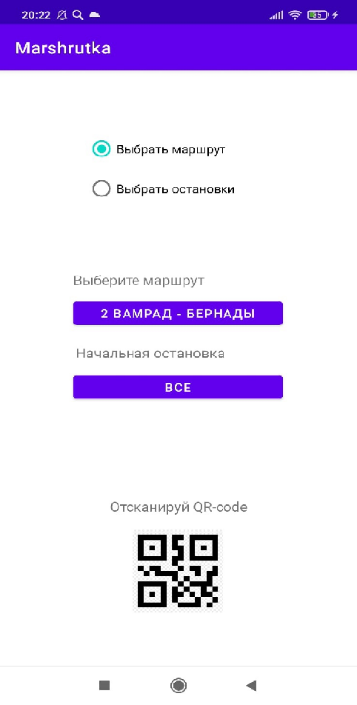


Рисунок 3.5 – главный экран приложения

Если вдруг пользователь допустил ошибку при выборе маршрута или остановки, то он может в любой момент изменить свой выбор повторно нажав на

кнопку (см. рисунок 3.6):

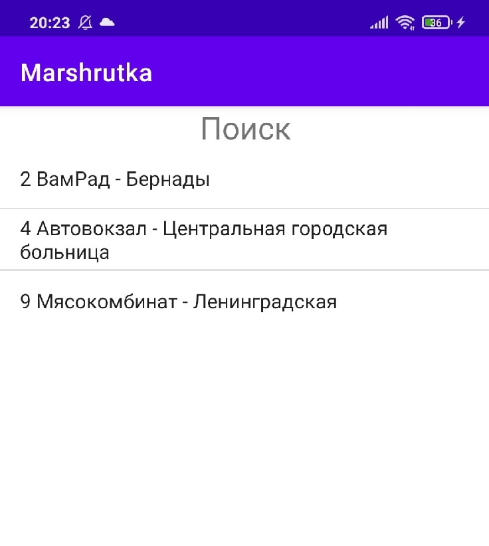


Рисунок 3.6 – экран выбора номера маршрута

Далее по желанию пользователь может выбрать начальную остановку вручную или же отсканировать её, нажав на QR-код внизу по центру экрана. При выборе сканирования остановки через QR-код у пользователя будет запрошено разрешение на использование камеры, после чего откроется сканер, представленный на рисунке 3.7:

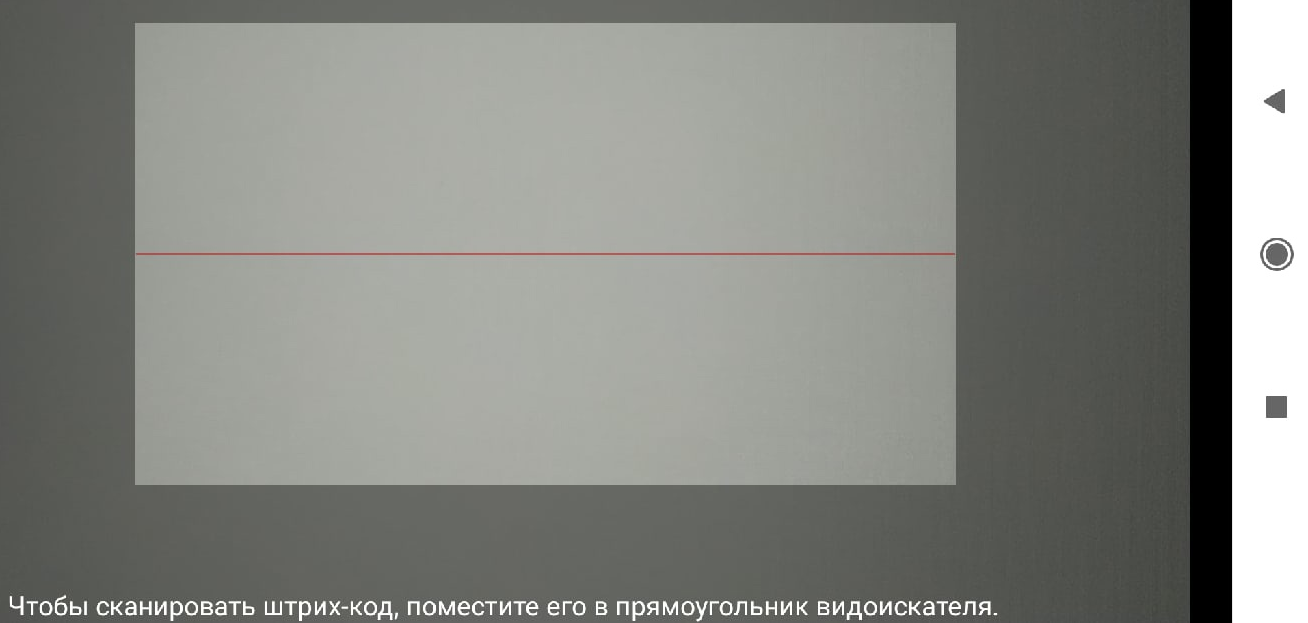


Рисунок 3.7 – сканер QR-кода

В случае, если недобросовестный пользователь попытается до выбора маршрута отсканировать начальную остановку, приложение выдаст уведомление с указанием на ошибку (см. рисунок 3.8) и не даст ему открыть сканер.

После выбора маршрута, начальной и конечной остановок пользователь видит диалоговое окно, в котором указаны введённые им данные заявки, с двумя вариантами ответа: отправить, отказаться. Такой вид отправки был сделан, чтобы пользователь смог в последний раз перепроверить введённые им данные и, в случае ошибки, отменить отправку и заполнить заявку заново корректные данными. Окно представлено на рисунке 3.9.



Рисунок 3.8 – уведомление об ошибке

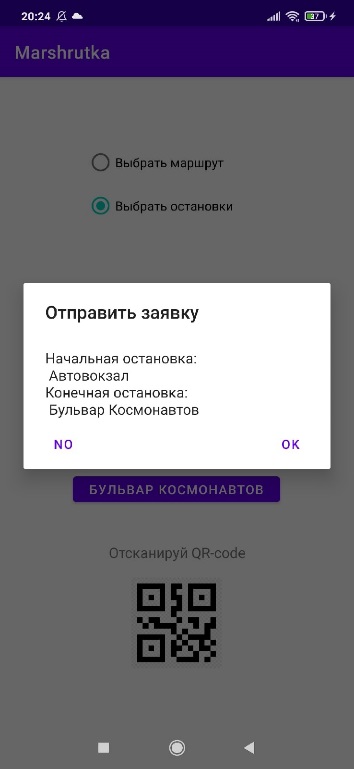


Рисунок 3.9 – отправка запроса

Если пользователь откажется отправлять заявку приложение вернёт его на главный экран и ему предстоит заполнить заявку заново с нуля (см. рисунок 3.10), иначе заявка отправляется на сервер и пользователю остаётся дождаться указанного им транспорта.

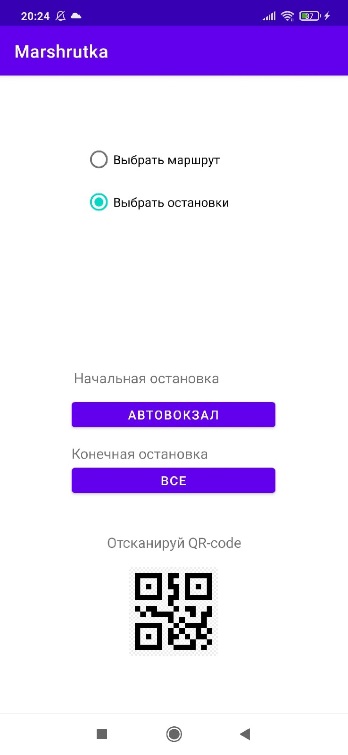


Рисунок 3.10 – пользователь отказался отправлять заявку

1. **ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ОБОСНОВАНИЯ**
   1. **Исходные данные**

Дипломная работа предусматривает разработку клиентской части Веб приложения автоматизированной системы управления маршрутным транспортом.

Уровень сложности программного продукта – средний. В ходе расчета экономической эффективности будет проведен расчет функций программного модуля и расчет полной себестоимости программного продукта.

Редактор исходного кода – Android Studio. Язык разработки программы – Java.

Определение общего объема программного обеспечения определяется исходя из количества и объема функций, реализуемых программой. При этом рассчитывается общий объем функций по каталогу, а также уточненный объем – это значение, скорректированное в соответствии с объемом функций разработанного программного обеспечения.

В разработке программной системе принял участие 1 человек – инженер- программист. Были выполнены следующие работы: проектирование архитектуры системы, реализация программного кода, тестирование системы.

Расчёты будут произведены в следующем порядке:

* расчёт объёма функций программного модуля;
* расчёт полной себестоимости программного модуля;
* расчет цены и чистой прибыли.
  1. **Расчет объема функций программного модуля**

Общий объем ПО (Vо) определим по формуле (4.1) исходя из количества и объема функций, реализуемых программой:

n

V0Vi, (4.1)

i1

где Vо – общий объем ПО;

Vi – объем отдельной функции ПО;

n – общее число функций.

Расчет общего объема ПО (количества строк исходного кода (LOC))

предполагает определение объема по каждой функции.

В том случае, когда на стадии технико-экономического обоснования проекта невозможно рассчитать точный объем функций, то данный объем может быть получен на основании ориентировочной (прогнозной) оценки имеющихся фактических данных по аналогичным проектам, выполненным ранее, или путем применения нормативов по каталогу функций.

Определение уточненного объема ПО.

На основании информации о функциях разрабатываемого программного обеспечения по каталогу функций определяется общий объем ПО в строках.

В зависимости от среды разработки, организационных и технологических условий производится корректировка объема на основе экспертных оценок.

Уточненный объем ПО (Vу) определяем по формуле (4.2):

, (4.2)

где *Vуi* – уточненный объем отдельной функции в строках исходного кода (LOC).

В таблице 4.1 приведен перечень и объем функций ПО.

Таблица 4.1 – Перечень и объем функций программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| Код функции | Наименование (содержание) функции | Объем функции строк исходного кода | |
| По каталогу Vi | Уточненный Vyi |
| 102 | Контроль, предварительная обработка и ввод информации | 490 | 250 |
| 107 | Организация ввода/вывода информации в интерактивном режиме | 280 | 180 |
| 203 | Обработка наборов и записей базы данных | 2370 | 620 |
| 305 | Формирование файла | 2130 | 150 |
| 506 | Обработка ошибочных и сбойных ситуаций | 1540 | 560 |
| 507 | Обеспечение интерфейса  между компонентами | 1680 | 800 |
| 703 | Расчет показателей | 420 | 300 |
| 707 | Графический вывод результатов | 420 | 260 |
| Итого | – | 9330 | 3120 |

Учитывая информацию, указанную в таблице 4.1, о функциях разрабатываемого программного обеспечения, уточненный объем ПО (*Vу*) составил 3120 строк исходного кода (LOC) вместо предполагаемого количества строк 9330.

* 1. **Расчет полной себестоимости программного продукта**

Стоимостная оценка программного средства у разработчика предполагает составление сметы затрат, которая включает следующие статьи расходов:

* + - заработную плату исполнителей (ΣЗПi);
    - отчисления на социальные нужды (ΣРсоцi);
    - материалы и комплектующие изделия (Рм);
    - спецоборудование (Рс);
    - машинное время (Рмв);
    - расходы на научные командировки (Рнк);
    - прочие прямые расходы (Рпр);
    - накладные расходы (Рнр);
    - затраты на освоение и сопровождение программного средства (Ро и Рсо).

Полная себестоимость (Сп) разработки программного продукта рассчитывается как сумма расходов по всем статьям с учётом рыночной стоимости аналогичных продуктов. Определяется по формуле (4.3):

Сп= ΣЗПi + ΣРсоцi + Рм + Рс + Рмв + Рнк + Рпр + Рнр + Ро + Рсо. (4.3)

Основной статьей расходов на создание ПП является заработная плата разработчиков (исполнителей) проекта (Токл + Нвд), в число которых принято включать инженеров-программистов, руководителей проекта, системных архитекторов, дизайнеров, разработчиков баз данных, Web-мастеров и других специалистов, необходимых для решения специальных задач в команде.

**Расчёт заработной платы**

Для расчета заработной платы разработчиков ПП необходимо определить:

* + - продолжительность времени разработки Фрв, которое устанавливается экспертным путем с учетом сложности, новизны ПП и фактически затраченного времени на схожих проектах;
    - количество разработчиков ПП.

Определим оклад исполнителя по формуле (4.4):

Токл = Тст1 р \* Тк / Nраб.дн. / мес. \* Фрв \* Кпр, (4.4)

где Тст1 р – месячная тарифная ставка рабочего 1 разряда;

Тк – тарифный коэффициент согласно разряду исполнителя;

Nраб.дн. / мес. – среднее количество рабочих дней в месяце;

Фрв – фонд рабочего времени исполнителя (продолжительность разработки в днях);

Кпр – коэффициент премий исполнителя.

Рассчитаем основную заработную плату инженера-программиста 2-й категории, используя формулу (4.4), на основе следующих данных:

* базовая месячная тарифная ставка рабочего 1 разряда, утверждённая на предприятии на дату написания дипломного проекта равна 195 белорусских рублей;
* среднее количество рабочих дней в месяце Nраб.дн. / мес. = 22 дня;
* коэффициент премирования будет установлен – 1,2;
* тарифный коэффициент 12 разряда инженера-программиста согласно тарифной сетке составляет Кт = 2,03;
* продолжительность разработки ПО определим Фрв = 30 дней.

Основная заработная плата программиста:

Токл = 195 \* 2,03 / 22 \* 30 \* 1,2 = 647,75 (белорусских рублей).

Надбавка за высокие достижения в труде (Нвд) – 25% от основной заработной платы, рассчитывается по формуле (4.5):

Нвд = Токл \* 25 / 100, (4.5)

Нвд = 647,75 \* 25 / 100 = 161,94 (белорусских рублей).

Заработная плата специалиста (ЗПспец) исполнителя программного продукта рассчитывается по формуле (4.6):

ЗПспец = Токл + Нвд. (4.6)

ЗПспец = 647,75 + 161,94 = 809,69 (белорусских рублей).

Результаты вычислений внесем в таблицу 4.2:

Таблица 4.2 – Расчёт заработной платы

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Категории работников** | **Раз ряд** | **Кт** | **Фэф** | **Кпр** | **Нвд** | **Зарплата, белорусские рубли** |
| **Заработная плата специалиста** |
| Инженер- программист  2-й категории | 12 | 2,02 | 30 | 1,2 | 25 | 809,69 |

Как видно из таблицы 4.2, затраты на оплату труда специалисту составят 809,69 белорусских рублей.

**Отчисления на социальные нужды.**

Отчисления на социальные нужды (Рсоц) определяется в соответствии с действующим законодательством по нормативу (34% – отчисления в ФСЗН + 0,6% отчисления по обязательному страхованию) и высчитываются по формуле (4.7):

Рсоц = ЗПспец \* 34,6 / 100. (4.7)

Рсоц = 809,69 \*34,6/100 = 280,15 (белорусских рублей).

**Спецоборудование.**

Расходы по статье «Спецоборудование» (Рс) включает все затраты на приобретение технических и программных средств специального назначения, необходимых для разработки ПМ, включая расходы на проектирование, изготовление, отладку и др.

Спецоборудование для написания программной системы приобретаться не будет, Рс = 0 (белорусских рублей).

**Материалы.**

По статье «Материалы и комплектующие изделия» (Рм) отражаются расходы на бумагу, картридж для принтера, необходимых для разработки ПП. Норма расхода материалов в суммарном выражении определяются в расчете на 100 строк исходного кода.

Сумма затрат на расходные материалы рассчитывается по формуле (4.8).

Рм = ЗПспец \* Нмз / 100, (4.8)

где Нмз – норма расхода материалов от основной заработной платы (принимаем 4%).

Рм = 809,69 \* 4 / 100 = 32,39 (белорусских рублей).

**Машинное время.**

Расход по статье «Машинное время» (Рмв) включают оплату машинного времени, необходимого для разработки и отладки ПМ. Они определяются в машино-часах по нормативам на 100 строк исходного кода машинного времени по формуле (4.9).

Рмв = Цм \* Vo / 100 \* Нмв, (4.9)

где Цм – цена одного машино-часа в белорусских рублях;

Vо – уточнённый общий объём функций строк исходного кода (LOC);

Нмв – норматив расхода машинного времени на отладку 100 строк кода, машино-часов.

Уточненный объем функций строк исходного кода (LOC) составил 3120. Цена одного машино-часа 0,7 руб. Норматив расхода машинного времени на отладку 100 строк кода – 0,7 машино-часов.

Рмв = 0,7 \* 3120 / 100 \* 0,7 = 15,29 (белорусских рублей).

**Научные командировки.**

В рамках разработки системы научные командировки не предусматриваются, Рнк = 0 (белорусских рублей).

**Прочие прямые затраты.**

Расходы по статье «Прочие прямые затраты» (Рпр) включают затраты на приобретение специальной научно-технической информации и специальной литературы и определяются в процентах к основной заработной плате исполнителей. Примем равными 10%. Следовательно, затраты высчитываются по формуле (4.10):

Рпр = ЗПспец \* 10/100. (4.10)

Рпр = 809,69 \* 10 / 100 = 80,97 (белорусских рублей).

**Накладные расходы.**

Затраты по статье «Накладные расходы» (Рнр) связаны с содержанием вспомогательных хозяйств, а также с расходами на общехозяйственные нужды. Определяются по нормативу в процентах к основной заработной плате исполнителей.

Затраты определены в размере – 50% от основной заработной платы исполнителя, так как Заказчик является бюджетным предприятием. Следовательно,

получаем следующую формулы (4.11):

Рнр = ЗПспец \* 0,5. (4.11)

Рнр = 809,69 \* 0,5 = 404,85(белорусских рублей).

Сумма вышеперечисленных расходов служит исходной базой для расчёта затрат на освоение и сопровождение программного модуля и рассчитывается по формуле (4.12):

Сумма затрат = ЗПспец + Рсоц + Рм +Рс +Рмв + Рнк + Рпр + Рнр. (4.12)

Сумма затрат = 809,69 + 280,15 + 32,39 + 0 + 15,29 + 0 + 80,97 + 404,85= =1623,34 (белорусских рублей).

**Затраты на освоение и сопровождение.**

Затраты на освоение ПП (Ро). Организация разработчик участвует в освоении ПП и несет соответствующие затраты, на которые составляется смета, оплачиваемая заказчиком по договору. Затраты на освоение определяются по установленному нормативу (Но = 10%) от суммы затрат по формуле (4.13):

Ро = Сумма затрат \* Но / 100, (4.13)

Ро = 1623,34 \* 10 / 100 = 162,33 (белорусских рублей).

Затраты на сопровождение (Рсо). Организация-разработчик осуществляет сопровождение ПП и несёт расходы, которые оплачиваются заказчиком в соответствии с договором и сметой на сопровождение. Затраты рассчитываются по формуле (4.14):

Рсо = Сумма затрат \* Нсо / 100, (4.14)

где Нсо – норматив расходов на сопровождение (примем Нсо = 10%).

Рсо = 1623,34 \* 10 / 100 = 162,33 (белорусских рублей).

Полная себестоимость программного продукта рассчитывается как сумма расходов по всем статьям по формуле (4.3):

Сп = 1623,34 + 162,33 + 162,33 = 1948 (белорусских рублей).

Результаты вычислений занесем в таблицу 4.3.

Таблица 4.3 – Расчет себестоимости программного модуля

|  |  |  |
| --- | --- | --- |
| Статьи затрат | Норматив | Затраты, белорусских рублей |
| Заработная плата всего | – | 809,69 |
| Отчисления на социальные нужды | 34,6% | 280,15 |
| Спецоборудование | – | 0 |
| Материалы | 5% | 32,39 |
| Машинное время | – | 15,29 |
| Научные командировки | – | 0 |
| Прочие затраты | – | 80,97 |
| Накладные расходы | 50% | 404,85 |
| Сумма затрат | – | 1623,34 |
| Затраты на освоение | 10% | 162,33 |
| Затраты на сопровождение | 10% | 162,33 |
| Полная себестоимость | – | 1948 |

Исходя из расчетов, полная себестоимость программного продукта составила 1948 белорусских рубля.

* 1. **Расчет цены реализации и чистой прибыли**

Для определения цены программного продукта необходимо рассчитать плановую прибыль. Плановая прибыль рассчитывается по формуле (4.15):

П = Сп \* R / 100, (4.15)

где Сп – полная себестоимость программного продукта;

R – уровень рентабельности ПМ (примем R = 30%).

Подставим исходные данные в формулу (4.15):

П = 1948\* 30 / 100 = 584,4 (белорусских рублей).

После расчета прибыли от реализации определяется прогнозируемая цена ПМ без налогов и рассчитывается по формуле (4.16):

Цп = Сп + П. (4.16)

Подставим исходные данные в формулу (4.16):

Цп = 1948 + 584,4 = 2532,4 (белорусских рублей).

Отпускная цена (цена реализации) ПМ включает налог на добавленную стоимость и рассчитывается по формуле (4.17):

Цо = Сп + П + НДС, (4.17)

где НДС – налог на добавленную стоимость, рассчитывается по формуле (4.18):

НДС = Цп \* НДС / 100, (4.18)

где НДС – налог на добавленную стоимость, в настоящее время равен 20%.

Рассчитаем НДС по формуле (4.18):

НДС = 2532,4 \* 20 / 100 = 506,48 (белорусских рублей).

Подставим исходные значения в формулу (4.17):

Цо = 2532,4 + 506,48 = 3038,88(белорусских рублей).

Прибыль от реализации ПМ за вычетом налога на прибыль (Пч) является чистой прибылью, остается организации разработчику и представляет собой экономический эффект от создания нового программного продукта. Рассчитывается по формуле (4.19).

Пч = П \* (1 - Нп / 100), (4.19)

где Нп – ставка налога на прибыль (в настоящее время Нп = 18%).

Подставим исходные данные в формулу (4.19):

Пч = 584,4 \* (1 – 18 / 100) = 479,21 (белорусских рублей).

Заносим полученные результаты в таблицу 4.4.

Таблица 4.4 – Расчет отпускной цены и прибыли ПМ

|  |  |  |
| --- | --- | --- |
| Наименование статей затрат | Норматив | Сумма затрат, руб. |
| 1 | 2 | 3 |
| Полная себестоимость | - | 1948 |
| Прибыль | 30 | 584,4 |
| Прогнозируемая цена без НДС | - | 2532,4 |
| НДС | 20 | 506,48 |
| Отпускная цена | - | 3038,88 |

Продолжение таблицы 4.4

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Наименование статей затрат | Норматив | Сумма затрат, руб. |
| Налог на прибыль | 18 | 105,19 |
| Чистая прибыль | - | 479,21 |

Таким образом, в ходе расчетов были определены основные экономические показатели:

• полная себестоимость – 1948 белорусских рублей;

• отпускная цена – 3038,88 белорусских рублей;

• чистая прибыль – 479,21 белорусских рублей.

При осуществлении массового выпуска программного обеспечения, затраты на разработку уменьшатся, что приведёт к уменьшению стоимости программного модуля и увеличению прибыли. Рассчитанная отпускная цена на программный продукт, разрабатываемый в данном дипломном проекте (без учета стоимости всей системы управления транспортом) так же будет уменьшена за счет универсальной модульной структуры и возможности подстроиться под требования заказчиков.

Если предположить, что цена за полноценный продукт (включая серверный модуль) вырастет не более, чем в 2 раза, получаем стоимость в 6000 белорусских рублей, что является допустимым показателем, в сравнении с аналогичными продуктами. При этом стоит учитывать, что функционал, предоставляемый разработанной системой шире и качественнее, чем у конкурентов.

Одно из неоспоримых преимуществ – отсутствие конкурентов, уже занявших большую часть рынка по Беларуси в области автоматизации маршрутного транспорта.

1. **ЭНЕРГОСБЕРЕЖЕНИЕ**

Энергосбережение – это организационная, научная, практическая, информационная деятельность государственных органов, юридических и физических лиц, направленная на снижение расходов (потерь) топливно-энергетических ресурсов в процессе их добычи, переработки, транспортировки, хранения, производства, использования и утилизации.

Энергосбережение с каждым годом становится все более актуальной проблемой. Ограниченность энергетических ресурсов, высокая стоимость энергии, негативное влияние на окружающую среду, связанное с ее производством, — все эти факторы приводят к альтернативе: разумнее снижать потребление энергии, нежели постоянно увеличивать ее производство.

Человечеством постепенно осознается необходимость перехода на экологически чистые источники энергии. На протяжении всего своего существования человечество использовало энергию, накопленную природой в течение миллиардов лет. При этом способы ее использования постоянно совершенствовались с целью получения максимальной эффективности. Все виды деятельности человека связаны с затратами энергии. Причиной же глобального пересмотра отношения к энергоресурсам является высокая энергоемкость продукции. Эта проблема в свою очередь влечет такие последствия как неэффективность экономики, неконкурентоспособность продукции, малая реализация на мировых и внутренних рынках, расходы на экспорт, остановка предприятий за ненадобностью производства дорогостоящей продукции.

Обеспечение ресурсосбережения – обязательное требование к технике, технологии, организации производства и непроизводственной деятельности, хозяйственному механизму. Новая техника должна требовать меньшего расхода ресурсов, как в процессе ее производства, так и в процессе эксплуатации. При решении проблем энергосбережения важно определить основные стратегические подходы и методы рационального использования энергоресурсов, которые могут быть как общими для всей экономики, так и специфичными для отдельных отраслей промышленности, сельского хозяйства и социальной сферы.

Среди наиболее общих подходов в стратегии энергосбережения можно назвать применение ресурсосберегающих технологий в сфере энерготехнологических объектов, использование методов математического моделирования и оптимизации при проектировании и реконструкции предприятий различных отраслей промышленности, замену дорогостоящих энергоемких видов энергоносителей, таких как электроэнергия на более дешевые, в частности, на природный газ, все более широкое использование возобновляемых источников энергии – ветра, солнца, биомассы.

В энергосбережении выделяют следующие группы мероприятий, обеспечивающие эффективное энергоиспользование и рациональное использование топливно-энергетических ресурсов: научно-технические, организационно- экономические, нормативно-технические, информационные и правовые. Научно-технические мероприятия по энергосбережению направлены на разработку и использование в производстве новых способов и устройств, отличающихся высокой энергоэффективностью. Организационные мероприятия по энергосбережению подразделяются на организационно-массовые и организационно-технические. Основным назначением организационно-массовой работы является доведение до всех членов трудового коллектива государственной важности экономного и бережного использования топлива и энергии, недопущения их потерь на всех участках производства.

Программы организационно-технических мероприятий по экономии топлива, тепловой и электрической энергии разрабатываются на всех уровнях управления и группируются по основным направлениям экономии применительно к производству продукции: совершенствование технологии производства, улучшение использования и структуры производственного оборудования, улучшение использования топлива и энергии в производстве, повышение качества сырья и применение менее энергоемких его видов.

Экономические меры по эффективному энергоиспользованию включают в себя систему гибких цен на энергоносители и универсальные тарифы, налоговую политику и меры материального стимулирования экономичного энергопотребления. К нормативно- техническим мероприятиям по энергосбережению относят действия по созданию соответствующих стандартов и других нормативно-технических и руководящих документов по обеспечению эффективного энергоиспользования и рационального использования топливно-энергетических ресурсов.

Информационные мероприятия по энергосбережению включают в себя проведение информационно-технических семинаров, выставок, конференций, симпозиумов по данной тематике, а также информирование населения через средства массовой информации об основных действиях по рациональному использованию энергии как на производстве, так и в быту. Тем не менее, современное предприятие невозможно представить без целого компьютерного парка. И перед нами становится задача обеспечить безотказную работу и исключить неэффективное использование компьютерной техники. Неэффективное использование компьютерной техники на предприятии может стать причиной необоснованного повышения затрат электроэнергии. Компьютер содержит ряд устройств, которые при работе потребляют значительную мощность. Таковыми являются мониторы, жесткие диски, системный контроллер, процессор и некоторые другие устройства. Производители аппаратного обеспечения большое внимание уделяют вопросам энергоэффективности своей продукции. Разрабатываются технологии для управления режимами электропитания и минимизации потерь электроэнергии, связанных с этим процессом.

Основное назначение любой системы управления питанием — автоматически переводить компьютер или отдельные его устройства в один из режимов пониженного энергопотребления. В свою очередь использование компьютеров на предприятии повышает эффективность во многих сферах деятельности, что в конечном итоге приводит к снижению затрат энергии и ресурсов. В частности, разработанный модуль позволяет сократить время решаемых задач инженерами, что приводит к сокращению потребления энергии.

Так как наибольший потенциал энергосбережения в сфере информационных технологий приходится именно на настольные компьютеры. Огромная проблема наблюдается когда настольные компьютеры простаивают включенными, это влечет колоссальные затраты на электро-энергию в мировых масштабах.

**ЗАКЛЮЧЕНИЕ**

За время работы над дипломным проектом было разработано приложение, реализующее часть необходимого функционала системы. Данное приложение является ориентированным на внедрение в реальную транспортную систему, которая позволит производить регистрацию пассажиров на перевозку, что, в свою очередь, позволит собирать статистику, определять маршруты с малой эффективностью и сократить экономические затраты на поддержание системы общественных перевозок.

Внедрение системы позволит увеличить прибыль перевозчиков, оптимизировать маршрутную сеть города, уменьшить расход энергоресурсов, повысить безопасность перевозок и качество обслуживания пассажиров. Таким образом, система обладает высоким потенциалом и может быть эффективно применена для решения задач управления транспортом.

Экономические расчеты показали жизнеспособность разработанного ПО для подсчета пассажиропотока и обосновали целесообразность создания системы и внедрения данного ПО.

Создание и продвижение системы представляется как полезным с социальной точки зрения, так и экономически эффективным.

**СПИСОК СОКРАЩЕНИЙ**

JWT – JSON Web Token.

JSON – JavaScript Object Notation.

LOC – строка исходного кода.

ПМ – программный модуль.

ОА – объект автоматизации.

ПП – программный продукт.

ПО – программное обеспечение.

**СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

* + - 1. Android Studio [Электронный ресурс]. – Режим доступа – https://developer.android.com – Дата доступа: 10.05.2021.
      2. Expo Bar Code Scanner [Электронный ресурс]. – Режим доступа – https://docs.expo.io/versions/latest/sdk/bar-code-scanner/. – Дата доступа: 09.05.2020.
      3. Java [Электронный ресурс]. – Режим доступа – https://www.java.com/en/ – Дата доступа: 13.05.2021.
      4. JavaScript Regular Expressions [Электронный ресурс]. – Режим доступа – https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Regular\_Expres sions/. – Дата доступа: 08.05.2020.
      5. Инструкция по работе с Android Studio [Электронный ресурс]. – Режим доступа – https://developer.android.com/guide – Дата доступа: 17.05.2020.
      6. Инструкция Java для android [Электронный ресурс]. – Режим доступа – https://developer.android.com/training/basics/firstapp – Дата доступа: 17.05.2020.