МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Разработка программного обеспечения
(направленность (профиль) / специализация)

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для доступа к веб-сайту»

Обучающийся О.А. Фабель _____
(Инициалы Фамилия) (личная подпись)

Руководитель канд. техн. наук Д.Г. Токарев
ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

## Аннотация

Актуальность темы, заключается в том, что все компании, большие и малые, пытаются перейти на автоматизированные системы. Облегчите жизнь себе и своим клиентам. Ведь упрощая свою работу, они получают положительных результатов в денежном выражении, так как удобство и простота использования важны для клиентов, а простота является очень важным фактором в их работе. У рынка мобильных технологий очень хорошее будущее, поскольку меняется не сам рынок, а потребители.

Цель работы заключается в теоретическом и практическом анализе по разработке мобильного приложения для доступа к веб-сайту для компании ООО «Гильдия мастерских».

Для достижения поставленной цели, необходимо будет решить следующие задачи:

- изучить организационную структуру предприятия с распределением ответственности, полномочий и взаимоотношений между работниками предприятия;
- построить функциональную модель и/или процессную модель организации «как есть», «как должно быть» в любых нотациях (IDEF, UML, ARIS, BPMN);
- описать и обосновать решения по системной архитектуре проекта;
- разработать мобильное приложение для доступа к веб-сайту;
- провести расчет экономической эффективности проекта.

Выпускная квалификационная работа состоит из введения, трех разделов, заключения, списка используемых источников.

Первый раздел работы включает в себя анализ объекта исследования. Представлен обоснованный выбор программных средств и технологий проектирования мобильного приложения.

Второй раздел подробно описывает процесс проектирования мобильного приложения. Также рассмотрены потоки данных, интерфейс пользователя, процесс взаимодействия пользователя с приложением.

Третий раздел посвящен технико-экономическому обоснованию программной реализации.

В заключение производится итоговый анализ о проделанной работе. Разработка мобильного приложения позволит оптимизировать работу компании ООО «Гильдия мастерских» и упростить обращение к web-сайту компании со стороны клиентов.

## Содержание

# Введение

Актуальность темы, заключается в том, что все компании, большие и малые, пытаются перейти на автоматизированные системы. Облегчите жизнь себе и своим клиентам. Ведь упрощая свою работу, они получают положительные результаты в денежном выражении, так как удобство и простота использования важны для клиентов, а простота является очень важным фактором в их работе. У рынка мобильных технологий очень хорошее будущее, поскольку меняется не сам рынок, а потребители [36].

Перспективы развития рынка мобильных технологий весьма благоприятны, поскольку меняется не столько сам рынок, сколько потребитель. Если вчера мечтой многих был безлимитный тариф, то сегодня с повсеместным распространением мобильного интернета прерогативы существенно изменились. С появлением на отечественном рынке 5G-решений произошло заметное перераспределение клиентов в пользу тех компаний, которые были готовы запустить эту услугу [15]. Современному владельцу мобильного устройства уже недостаточно просто общаться, ему необходим круглосуточный доступ к глобальной сети, с помощью которой можно не только найти нужную информацию, но и связаться с друзьями, сообщить свое местоположение и многое другое.

Данная работа направленна на разработку программы (мобильного приложения) для компании – ООО «Гильдия мастерских», которая упростит работу владельцу.

Объектом исследования является компания ООО «Гильдия мастерских».

Предметом исследования является методика разработки мобильных приложений.

Цель работы заключается в теоретическом и практическом анализе, и разработке мобильного приложения для доступа к веб-сайту для компании ООО «Гильдия мастерских».

Для достижения поставленной цели, необходимо будет решить следующие задачи:

- изучить организационную структуру предприятия с распределением ответственности, полномочий и взаимоотношений между работниками предприятия;
- построить функциональную модель и/или процессную модель организации «как есть», «как должно быть» в любых нотациях (IDEF, UML, ARIS, BPMN);
- описать и обосновать решения по системной архитектуре проекта;
- разработать мобильное приложение для доступа к веб-сайту;
- провести расчет экономической эффективности проекта.

# 1 Анализ задачи и выбор средств разработки

## 1.1 Характеристика предприятия и его структура

ООО «Гильдия мастерских» занимается исследование, разработкой программного обеспечения различной сложности. Находится по адресу: 350005, Краснодарский край, г.Краснодар, ул.Волгоградская, д. 123.

Руководитель: Аристов Иван Иванович. В должности руководителя с 28.11.2022, по данным ЕГРЮЛ/Росстат.

Основным видом деятельности ООО «Гильдия мастерских» по ОКВЭД является 62.01 Разработка компьютерного программного обеспечения.

Организационную структуру организации можно увидеть на рисунке 1.


Рисунок 1 – Организационная структура ООО «Гильдия мастерских»

«Каждое подразделение выполняет свои функции в деятельности компании:

- руководство компании: включает в себя генерального директора (CEO), который отвечает за общее управление компанией, принятие стратегических решений и развитие бизнеса;

– отдел продаж: занимается продажами и маркетингом, поиском новых клиентов и развитием бизнеса;
– отдел разработки: разрабатывает программные продукты;
– отдел технической поддержки: предоставляет информационно-технологическое сопровождение. Отдел управления проектами: занимается планированием, координацией и управлением проектами, в том числе распределением ресурсов, управлением бюджетом и сроками;
– отдел качества: отвечает за контроль качества продуктов и услуг, тестирование и анализ результатов, а также разработку и внедрение методик и стандартов;
– отдел управления персоналом: занимается управлением кадрами, включая подбор и найм сотрудников, оценку и развитие персонала, а также создание политик и процедур по управлению персоналом;
– финансовый отдел: отвечает за управление финансами компании, включая учет и отчетность, бюджетирование, финансовый анализ и планирование;
– юридический отдел: занимается юридическими вопросами, связанными с бизнес-операциями компании, включая согласование договоров, защиту прав и интересов компании, обеспечение соответствия законодательству и т.д.» [1].

Проведем ознакомление с web-сайтом компании (рисунки 2-3).

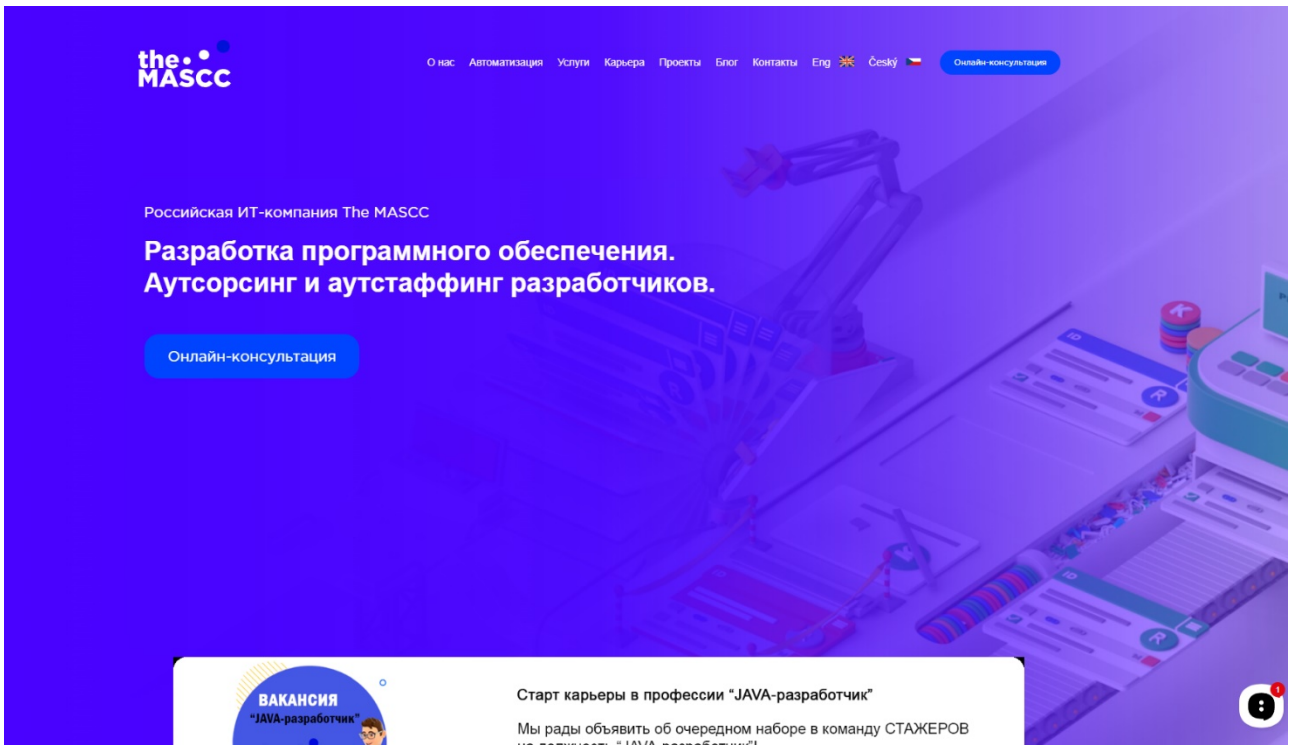



Рисунок 2 – Главная страница сайта компании

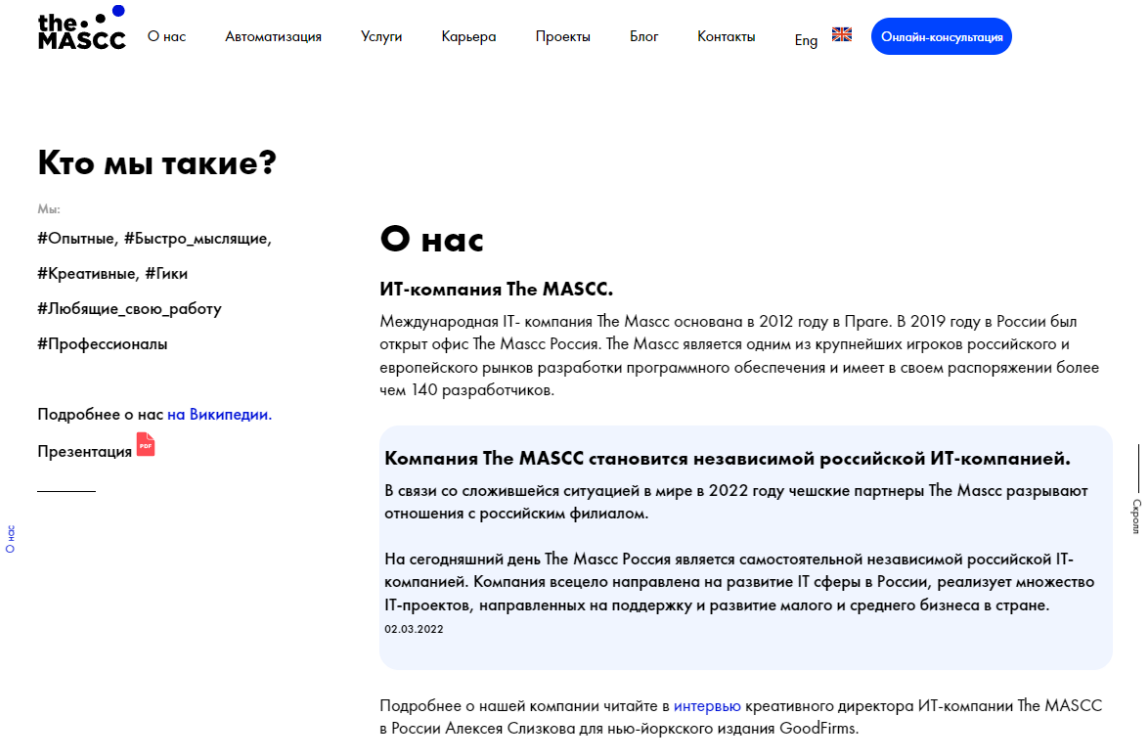



Рисунок 3 – Информация о компании на официальном сайте

Для удобства доступа к web-сайту компании, будет реализовываться мобильное приложение, чтобы было комфортное использование ресурса без громоздких страниц браузера на мобильном устройстве.

## 1.2 Актуальность данной работы и ее обоснование

Спрос на мобильные устройства с каждым днем набирает обороты, а соответственно и растет спрос на приложения, без которых не обходится работа на гаджетах [16]. Согласно статистике продаж мобильных устройств (смартфонов) в России по сравнению с 2021 годом в 2022 году количество проданной продукции увеличилось на 9 млн. устройств. Но опираясь на данные по первому кварталу 2025 года, упали на 26% в штуках, до 5,5 млн устройств, из-за сложных экономических ситуаций в стране [8]. Данный факт можно рассмотреть на графиках, представленных на рисунке 4.

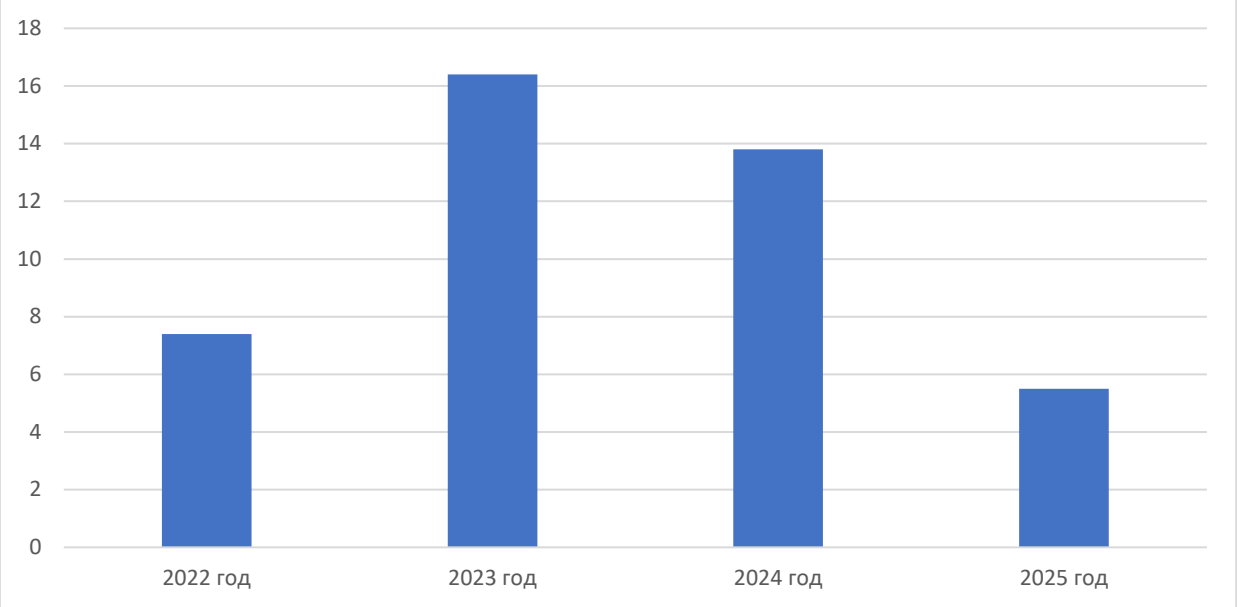



Рисунок 4 – График продаж мобильных устройств в России за период с 2022 по 2025 гг. (в мл. единиц)

Но, несмотря на снижение в потребительской динамике покупок мобильных устройств, спрос на пользование мобильных браузеров для

просмотра необходимых запросов остается неизменным, и он будет всегда и будет расти [17].

Для наглядности посмотрим динамику роста пользования мобильного приложения для просмотра сайтов с мобильных устройств за 2022 – 2025 гг. (рисунок 5).

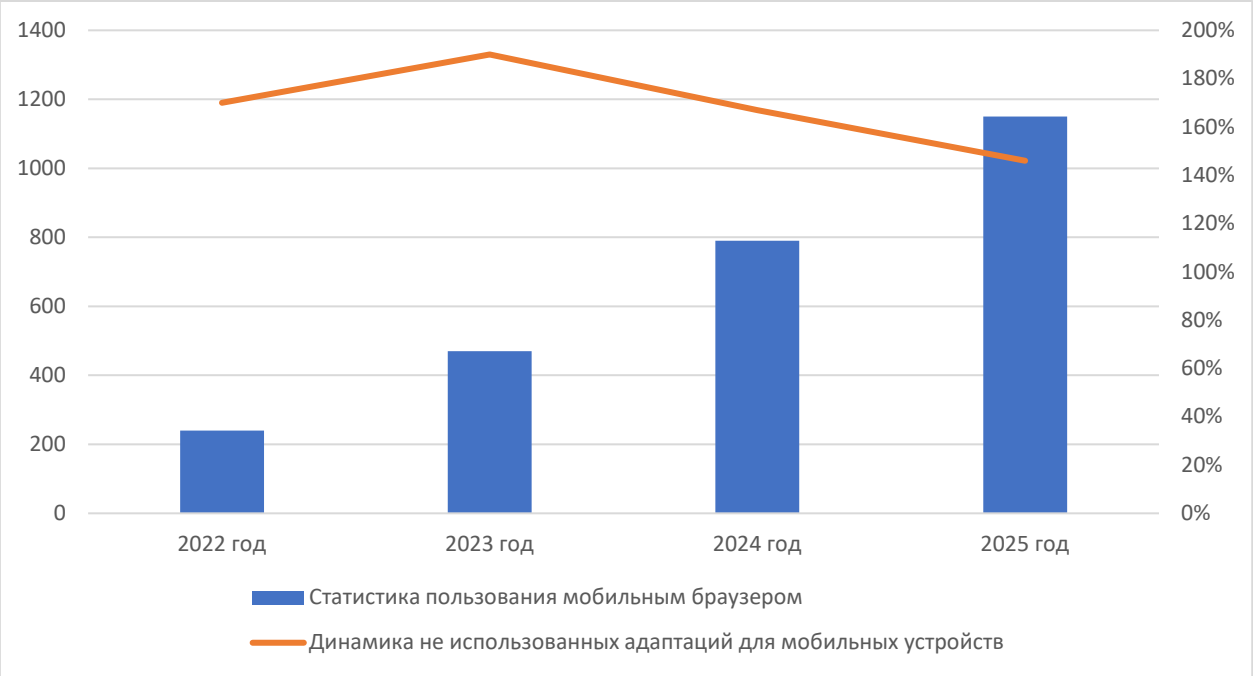



Рисунок 5 – График динамики пользования мобильных браузеров

Опираясь на данные показанные на рисунке 5, можно заметить, что пользование мобильным браузером для просмотра необходимых данных на сайтах стабильно в спросе, но что касается вопроса, касаемого адаптации сайтов компаний под мобильные устройства, не все к этому прибегают.

Таким образом, для удобства пользования web-сайтом компании ООО «Гильдия мастерских» на мобильных устройствах, необходимо разработать мобильное приложение, которое позволит эффективно интегрировать информацию с сайтом компании.

## 1.3 Выбор средств и технологий разработки

В настоящее время технологии стремительно развиваются и все больше интегрируются в нашу повседневную жизнь. Трудно представить себе человека, полностью защищенного от современных гаджетов [2].

Они уделяют большое внимание личной безопасности и конфиденциальности данных. Появляются всевозможные функции безопасности: сканеры отпечатков пальцев, сканеры сетчатки глаза, распознавание голоса, поверхности, чувствительные к давлению, и многое другое.

Одним из флагманских продуктов Google является ОС Android, которая установлена на большинстве устройств. Компания имеет огромное количество идей по улучшению технологий и тратит на это огромные средства, чтобы конечный пользователь мог эффективно и комфортно использовать мобильное устройство [2], [5], [16], [17], [28].

Разработка приложений для Android – это процесс создания мобильных продуктов для устройств, работающих под управлением операционной системы Android. Для этого используется официальная среда разработки Android Studio, языки программирования Kotlin и Java, а также мощные инструменты SDK (Software Development Kit) [21].

Приложение Android использует в своей работе окна на каждое действие. Каждое окно имеет свою уникальную жизнь и индивидуальность.

Жизненный цикл приложения Android показан на рисунке 6.

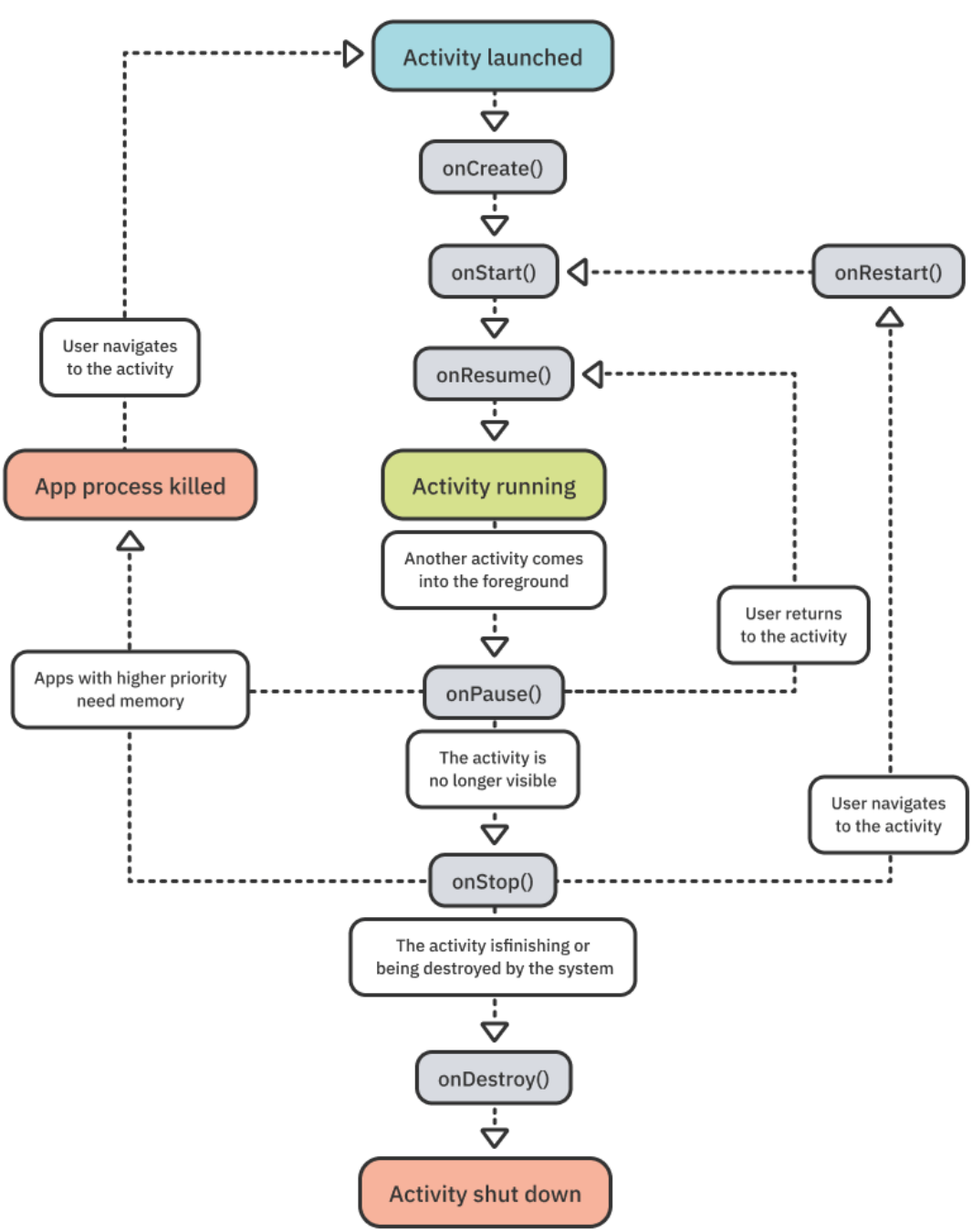



Рисунок 6 – Жизненный цикл приложения Android [25]

«Описание основных моментов жизненного цикла приложения:

– onCreate(): действие переходит в состояние Создано. Здесь вы выполняете логику, которая должна выполняться только один раз за всё время существования действия. Это может включать в себя настройку представления содержимого, привязку действия к

ViewModel, создание экземпляров некоторых переменных в области видимости класса и т. д.;

- onStart(): действие переходит в состояние Started. Этот вызов делает действие видимым для пользователя, поскольку приложение готовится к тому, чтобы действие вышло на передний план и стало интерактивным [26];
- onResume(): действие переходит в состояние возобновления. Теперь пользователь может взаимодействовать с действием. Здесь вы можете включить любую функцию, которая должна выполняться, пока компонент виден и находится на переднем плане [28];
- onPause(): действие переходит в состояние приостановлено. Этот вызов указывает на то, что действие больше не находится на переднем плане, но может быть видимым, например, если пользователь работает в режиме нескольких окон. В это время следует приостановить или скорректировать операции, которые не должны выполняться или должны выполняться с осторожностью. Действие остается в этом состоянии до тех пор, пока не возобновится, например, при открытии или закрытии нижнего листа в действии, или пока не станет полностью невидимым для пользователя, например, при открытии другого действия [6];
- onStop(): действие переходит в состояние остановлено. Действие больше не отображается пользователю. Здесь вы должны освободить или настроить ресурсы, которые не нужны, пока действие не отображается пользователю. Вы также можете воспользоваться этой возможностью, чтобы завершить задачи, которые требуют значительных вычислительных ресурсов, например, операции с базой данных [24];
- onDestroy(): действие переходит в состояние Destroyed. Здесь действие завершается» [4].

При изучении жизненного цикла приложения на ОС Android, перейдем к сравнительному анализу программных инструментов для разработки приложений для данной системы. [23]-[26]

Android Studio – официальная интегрированная среда разработки (IDE) для разработки приложений Android. [7]-[8] Она подходит для взаимодействия на языках Java и Kotlin. С её помощью разработчики создают приложения для мобильных, планшетов, телевизоров, часов и других устройств [9].

Рабочая область Android Studio представлена на рисунке 7.

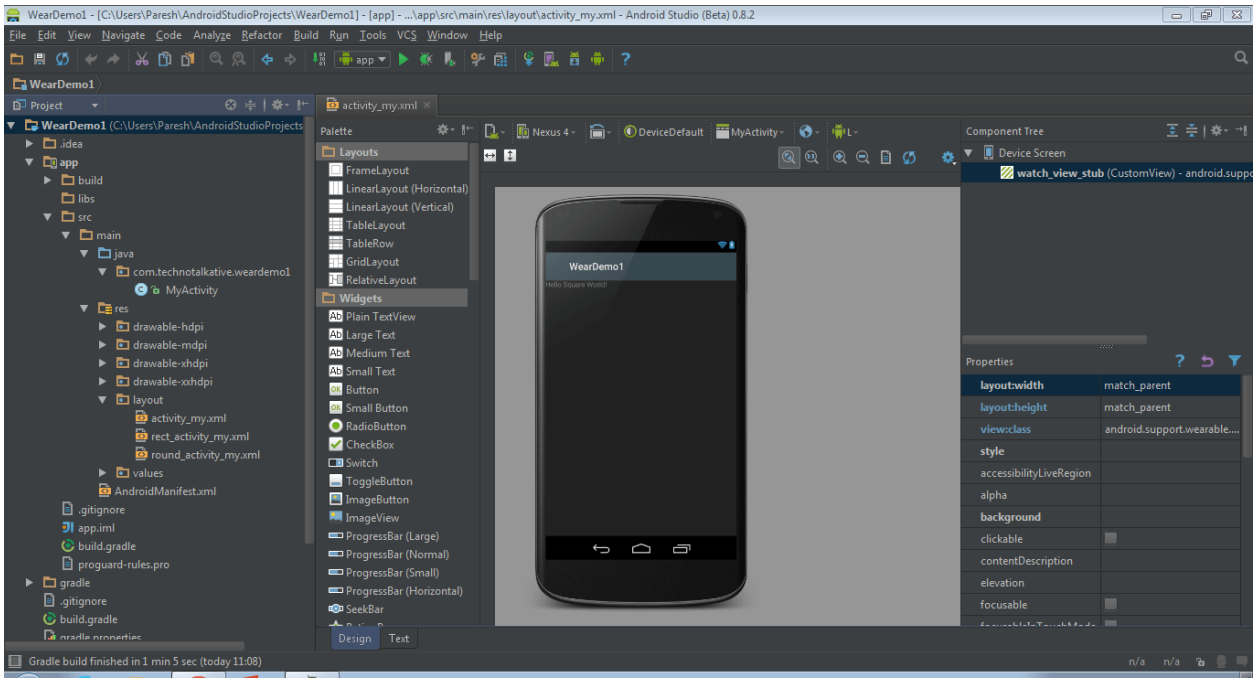



Рисунок 7 – Рабочая область Android Studio

«IDE содержит инструменты для разработки, отладки, тестирования и отслеживания производительности приложений. Android Studio – бесплатная, работает на Windows, Mac и Linux. Приложения можно сразу публиковать в магазине Google Play» [5].

Flutter – это открытый фреймворк разработки мобильных приложений, созданный компанией Google [23]. Этот инструмент позволяет разработчикам создавать кроссплатформенные приложения, используя один и тот же код для

iOS и Android. Flutter основан на языке программирования Dart, который также разрабатывается Google.

Рабочая область Flutter представлена на рисунке 8.

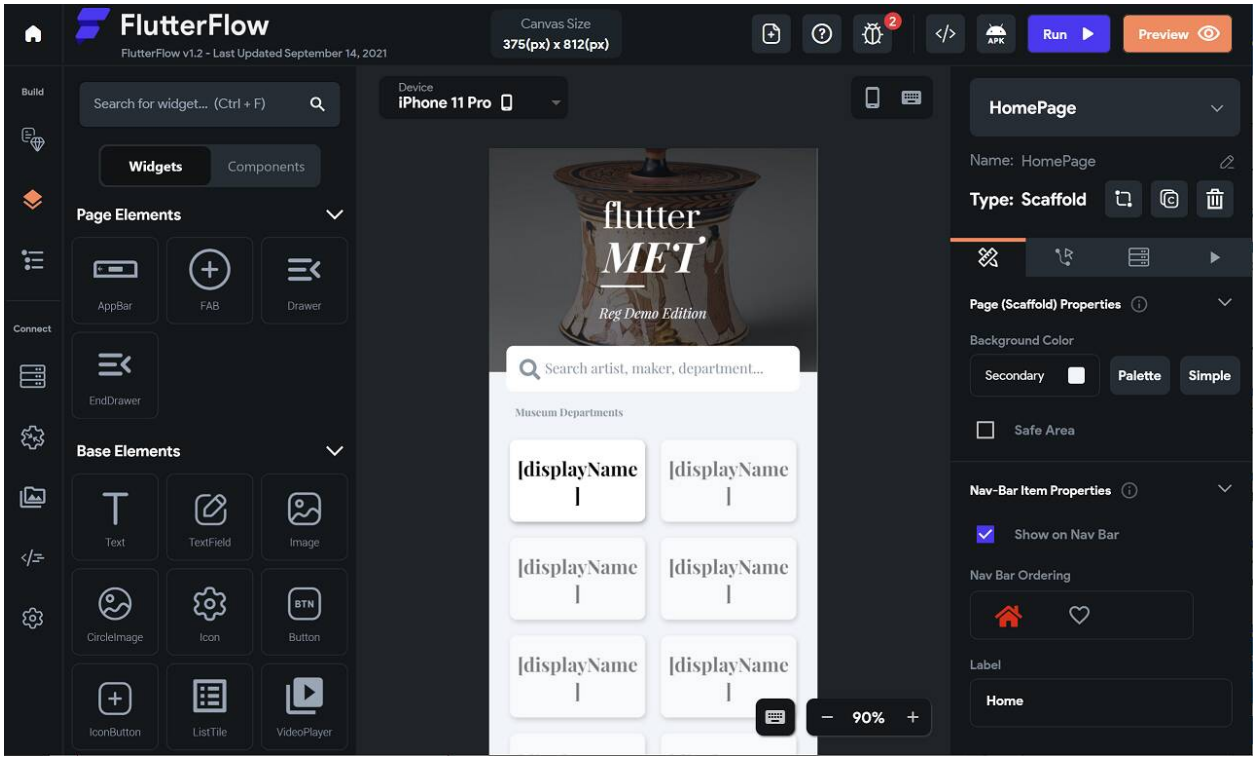



Рисунок 8 – Рабочая область Flutter

Фреймворк Flutter «стал популярным среди разработчиков благодаря своей универсальности и высокой производительности. Он предлагает широкий набор готовых виджетов и инструментов для создания красивых и функциональных приложений. Благодаря горячей перезагрузке, разработчики могут быстро видеть изменения в своем приложении и мгновенно тестировать новый код» [5].

«Flutter активно поддерживается сообществом разработчиков и постоянно обновляется, что делает его одним из самых перспективных фреймворков для создания мобильных приложений. Он позволяет ускорить процесс разработки и сэкономить время, благодаря чему становится все более популярным выбором для компаний и индивидуальных разработчиков» [13].

Опираясь на основе изученных двух программных областей, в которых можно производить разработку мобильного приложения, более эффективным будет являться Android Studio. [30]-[40]

«Программа предоставляет множество шаблонов для распространенных элементов программирования для операционной системы Android, а также полезные, удобные и справочные материалы. Окно программы отображает структуру созданного проекта, что делает процесс написания программы более удобным и гибким. Более того, все изменения, вносимые в программу, отображаются в режиме реального времени, поэтому вы можете сразу оценить ее эффективность и производительность» [5].

Многоцелевая среда разработки Android Studio «включает в себя полный набор всех необходимых инструментов и функций, благодаря которым такое трудоемкое и сложное создание Android-приложения становится максимально простым и интересным. При этом программа работает надежно и стабильно, проста в освоении и подойдет как профессиональным разработчикам, так и новичкам в программировании приложений для ОС Android. Разработка приложений для Android не только информативна, но и упрощает внедрение новых услуг для клиентов» [22].

Основным языком программирования на Android является Java [7]. Для создания разметки приложения и элементов интерфейса используется язык разметки XML. Писать программы для Android на Java можно практически в любой среде программирования, но есть официальная среда разработки – Android Studio [3].

«Основные возможности языка программирования Java:

- автоматический контроль отзыва;
- расширенные возможности для устранения неравенства;
- разработанные языковые средства для многопоточных приложений;
- унифицированный доступ к базам данных на основе JDBC» [30].

Название Java дано не только самому языку, но и платформе для разработки приложений корпоративного уровня, созданных на этом языке.

«Java – язык, используемый для разработки мобильных приложений для операционной системы Android» [9].

В ходе выполнения аналитической части проекта, для достижения поставленной цели были выполнены задачи по изучению организационной структуры организации, проанализированы существующие в ней бизнес-процессы, четко определен предмет автоматизации.

На основе проведенного анализа был произведен обоснованный выбор программных средств и технологий, в качестве языка для программной реализации был выбран – Java, более эффективным IDE будет являться Android Studio.

В заключение, можно сказать, что анализ бизнес-процессов объекта автоматизации и выбор средств разработки выявляют основные цели и задачи проекта, а также служат основой для успешной реализации дальнейших этапов проектирования и разработки.

# 2 Программная реализация мобильного приложения для доступа к веб-сайту компании ООО «Гильдия мастерских»

## 2.1 Разработка функциональной модели мобильного приложения

Построение функциональной модели необходимо для того, что перед разработкой системы заказчик и разработчик могли ясно представить, какие функциональные возможности будут заложены в систему [10].

Рассмотрим контекстную диаграмму IDF0 «как есть» и «как должно быть»


Рисунок 9 – IDEF0 «как есть»

Как видно из рисунка 9, производится обычный доступ к сайту компании без использования приложений, что затрудняет эффективно интегрировать информацию с сайта компании. Для этого целесообразно ввести в процесс работы с сайтом компании, разрабатываемое приложение (рисунок 10).


Рисунок 10 – IDEF0 «Как должно быть»

В ходе анализа проектируемой ИС было определено 2 актера:

– администратор (тот, кто управляет приложением);

– гость (пользователь).

На рисунке 11 изображена контекстная диаграмма вариантов использования проектируемого интернет-магазина. Контекстная диаграмма представляет собой граф, узлами которого являются достаточно крупные блоки функциональности системы.

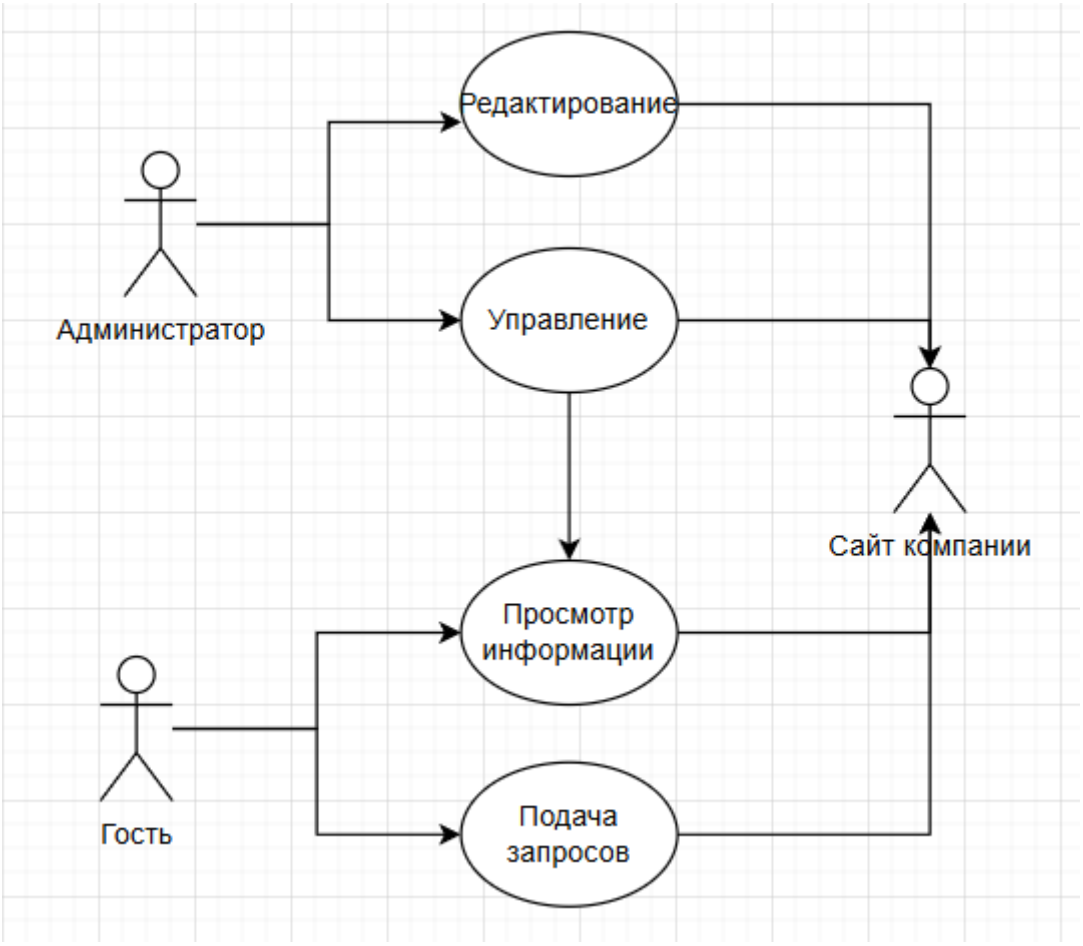



Рисунок 11 – Контекстная диаграмма вариантов использования

На основе контекстной диаграммы были созданы несколько диаграмм декомпозиции. Такие диаграммы, как правило, представляют собой «ромашку», в центре которой декомпозируемый вариант использования, а вокруг – входящие в него обязательные или расширяющие составные части. В работе представлены две диаграммы декомпозиции для разных вариантов использования [6].

Диаграмма декомпозиции варианта использования «Онлайн консультация», представленная на рисунке 12, отображает взаимодействие двух актеров – Пользователя и Администратора.

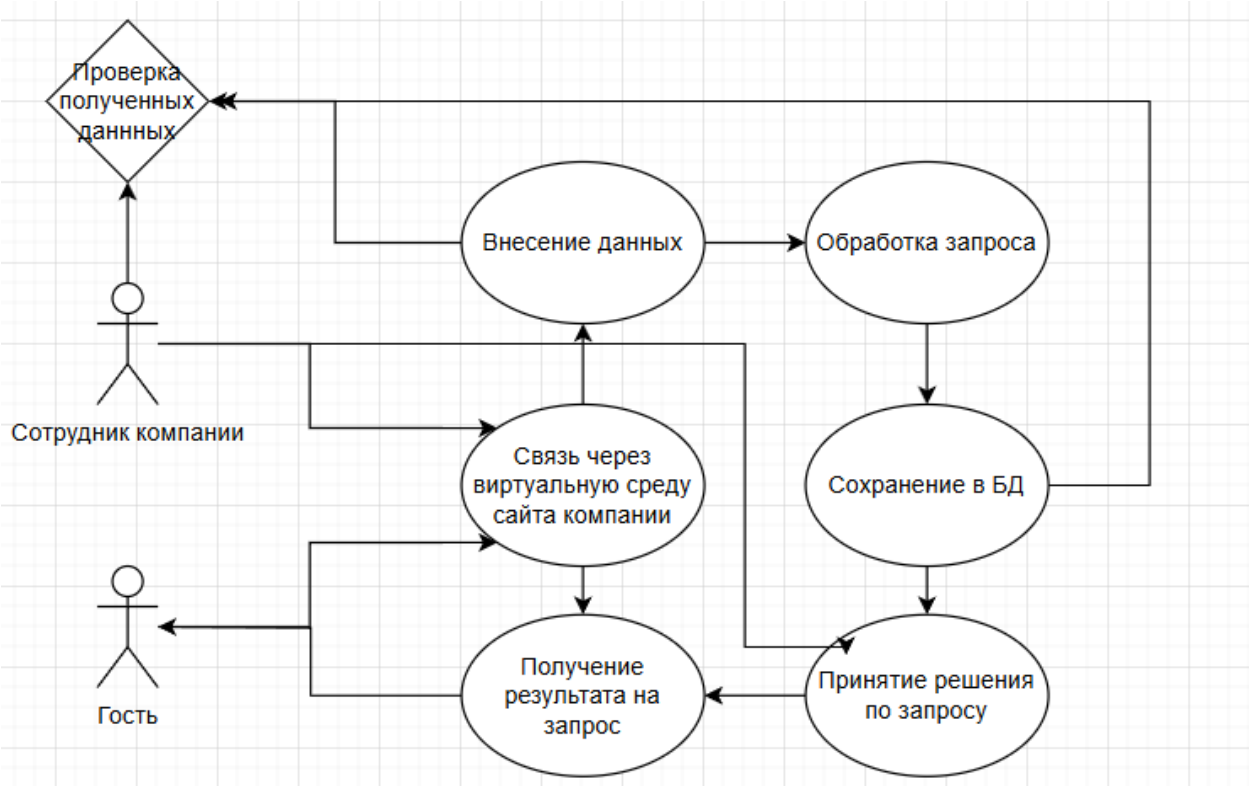



Рисунок 12 – Диаграмма декомпозиции варианта использования «Онлайн консультация» предоставленной на сайте компании

Диаграмма декомпозиции для варианта использования «Просмотр сайта компании», представленная на рисунке 13, описывает процесс просмотра сайта компании. В качестве актеров задействованы «Гость».


Рисунок 13 – Диаграмма декомпозиции для варианта использования «Просмотр сайта компании»

После создания необходимых диаграмм вариантов использования, проработайте их, основной целью чего будет определение того, при каком поведении система будет предоставлять требуемую функциональность.

Диаграмма автомата – это тип диаграммы, позволяющий детализировать варианты использования [19].

Диаграммы автоматов – один из способов подробного описания поведения в UML на основе явной идентификации состояний и описания переходов между состояниями.

Диаграммы автоматов используются для моделирования динамических аспектов системы. Они помогают моделировать жизненный цикл объекта и используются для описания поведения, реализованного в варианте использования, или поведения экземпляра сущности (класса, объекта, компонента, узла или всей системы).

Граф автомата – это особый тип графа, представляющий определенный автомат.

Вершины графа – возможные состояния машины, представленные соответствующими графическими символами, а дуги – переходы из одного состояния в другое.

Диаграммы состояний могут быть вложенными, чтобы обеспечить более детальное представление отдельных элементов модели.

В UML состояние – это абстрактный объект, моделирующий конкретную ситуацию, в которой выполняются определенные условия [27].

Разработанная информационная система предоставляет четыре категории пользователей с различными функциями и разрешениями, и соответствующим образом реализует пользовательский интерфейс.

На рисунке 14 показана контекстная диаграмма для программы, которая представляет собой определенную иерархию пунктов меню и диалоговых окон, которые может выбирать пользователь.

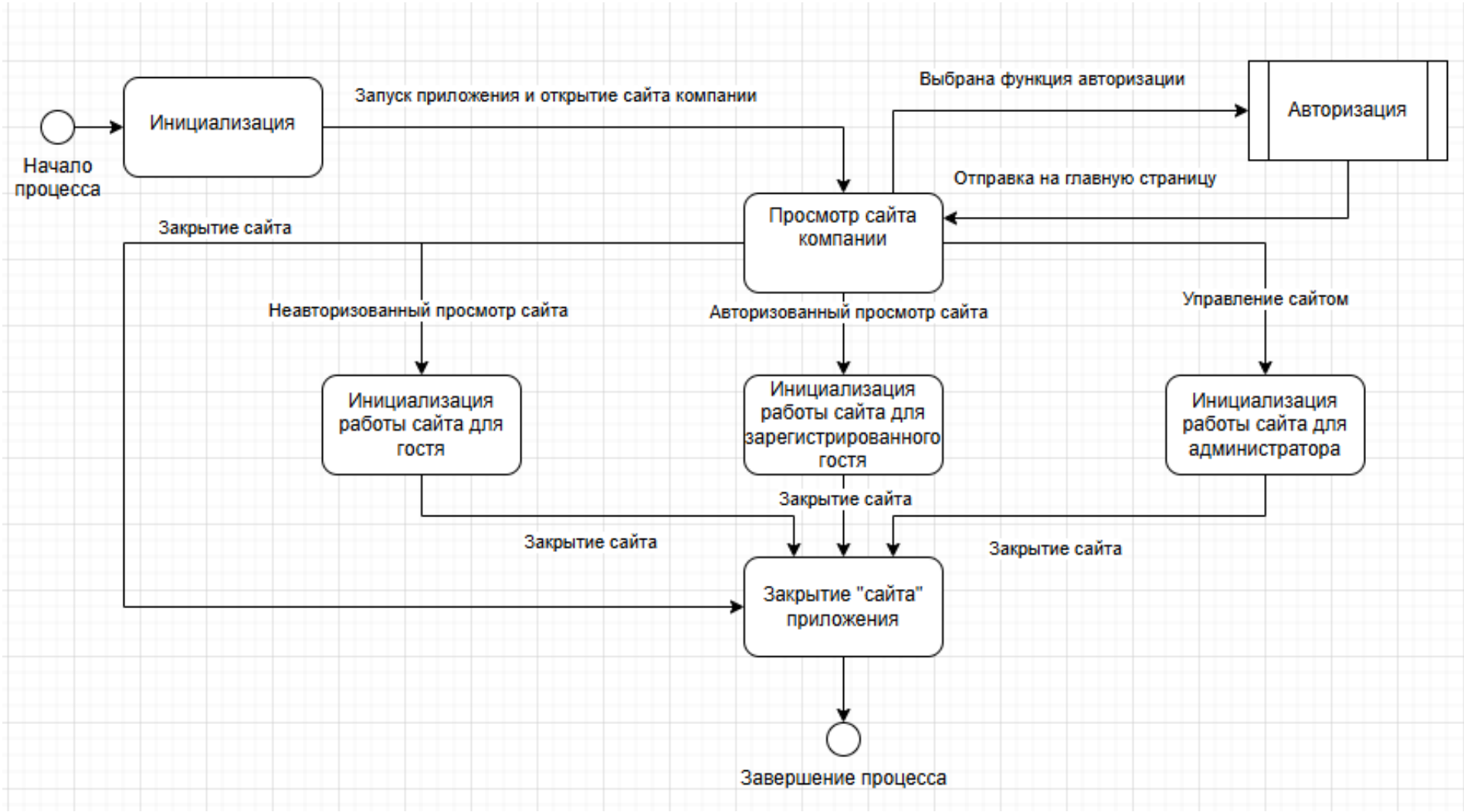



Рисунок 14 – Контекстная диаграмма приложения

После успешной аутентификации пользователя инициализируются определенные подсистемы на основе предоставленных пользователю прав доступа.

Рассмотрим схему приложения для подсистемы «авторизация» (рисунок 15).

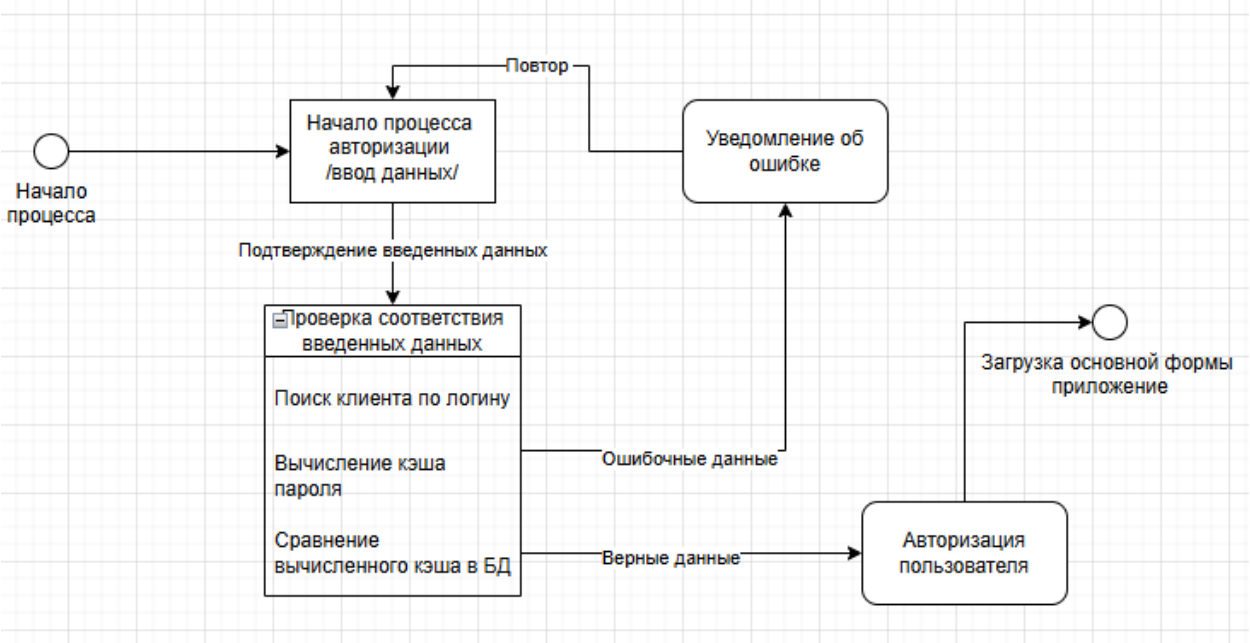



Рисунок 15 – Диаграмма автоматов для подсистемы «Авторизация»

Пользователи принимают участие. «Пользователь – физическое лицо или организация, использующие операционную систему для выполнения определенной функции. Для входа в систему вам необходимо ввести свое имя пользователя и пароль. Далее система проверяет совпадение введенного имени пользователя и пароля. Если пользователь вводит данные правильно, он авторизуется и получает права. В противном случае будет выведено сообщение о неверно введенной паре логин/пароль и количестве оставшихся попыток входа. Далее пользователю необходимо ввести данные заново. У вас есть три попытки входа в систему» [15]. Если максимальное количество попыток ввода пароля будет превышено, пользователь будет заблокирован, а приложение принудительно закрыто.

## 2.2 Разработка информационной модели

Следующим этапом проекта веб-приложения является создание информационной модели. «Эта модель представляет отношения между элементами системы с точки зрения структур данных. То есть создание модели данных сводится к созданию диаграммы классов анализа, диаграммы классов базы данных и диаграмм классов приложений» [12].

Диаграмма классов анализа.

«Основными понятиями объектно-ориентированного программирования являются понятия объектов и классов, которые представляются как абстракции реальной или воображаемой сущности (множества сущностей). Класс анализа – это более абстрактная сущность, состоящая из одного класса и набора из одного или нескольких классов. Таким образом, анализ классов представляет собой грубозернистую абстракцию, описывающую фрагмент системы на концептуальном уровне (без точного определения атрибутов и операций)» [15].

«Диаграмма классов анализа по сути является прототипом классической диаграммы классов. Элементы, показанные на диаграмме, – это классы и отношения между ними. В этом случае для отображения классов можно использовать стандартные именованные классы (прямоугольник) с соответствующим стереотипом или значком стереотипа» [15].

«Диаграмма классов – это диаграмма, вершинами которой являются различные типы классов анализа (управление, граница, класс сущностей), соединенные различными типами структурных отношений» [39].

На рисунке 16 показана диаграмма классов анализа проектируемой системы. На рисунке основное внимание уделено граничным классам, которые представляют структуру пользовательского интерфейса, и классам сущностей, которые представляют структуру базы данных.

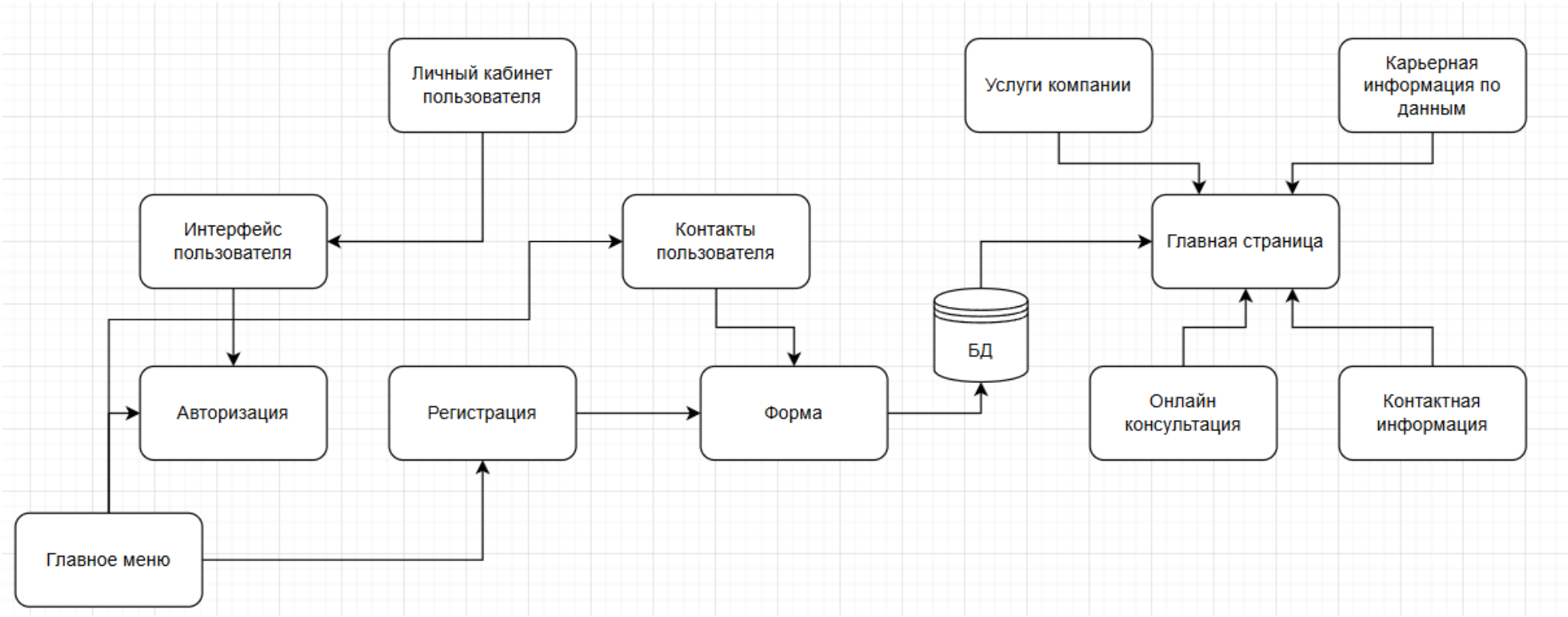



Рисунок 16 – Информационная модель классов

## 2.3 Разработка компонентной модели

На завершающем этапе построения информационной системы интернет-хранилища строительных материалов устанавливается распределение функций по компонентам и осуществляется распределение по узлам системы. Приведены схемы компонентов и реализаций.

«Диаграмма компонентов описывает физическое представление системы. Позволяет определить архитектуру встроенной системы путем установления зависимостей между программными компонентами, которыми могут быть исходный и исполняемый код. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними» [18].

«Схема компонентов предназначена для следующих целей:

- представление общей структуры исходного кода программной системы;
- характеристики прикладной версии программного комплекса;
- учет многократного использования отдельных фрагментов программного кода;
- представление концептуальных и физических схем баз данных» [29].

В разработке компонентных диаграмм участвуют как системные аналитики, так и архитекторы и программисты. Диаграмма компонентов обеспечивает непрерывный переход от логического представления проекта к его конкретной реализации в программном коде.

В UML существует три типа ядер:

- «развертывания, которые обеспечивают непосредственное выполнение системой своих функций (такими компонентами могут быть динамически подключаемые библиотеки с расширением dll);
- рабочие продукты – это файлы с исходными текстами программ;

– исполнения, представляющие собой исполняемые модули (файлы с расширением exe)» [20].

Другим способом спецификации различных видов компонентов является явное указание его стереотипа перед именем. «В языке UML для компонентов определены следующие стереотипы:

– «file» – любой файл, кроме таблицы:
– «executable» – программа (исполняемый файл);
– «library» – статическая или динамическая библиотека;
– «source» – файл с исходным текстом программы;
– «document» – остальные файлы (например, файл справки);
– «table» – таблица базы данных» [11].

Составим диаграмму компонентов для информационной модели сайта компании (рисунок 17).

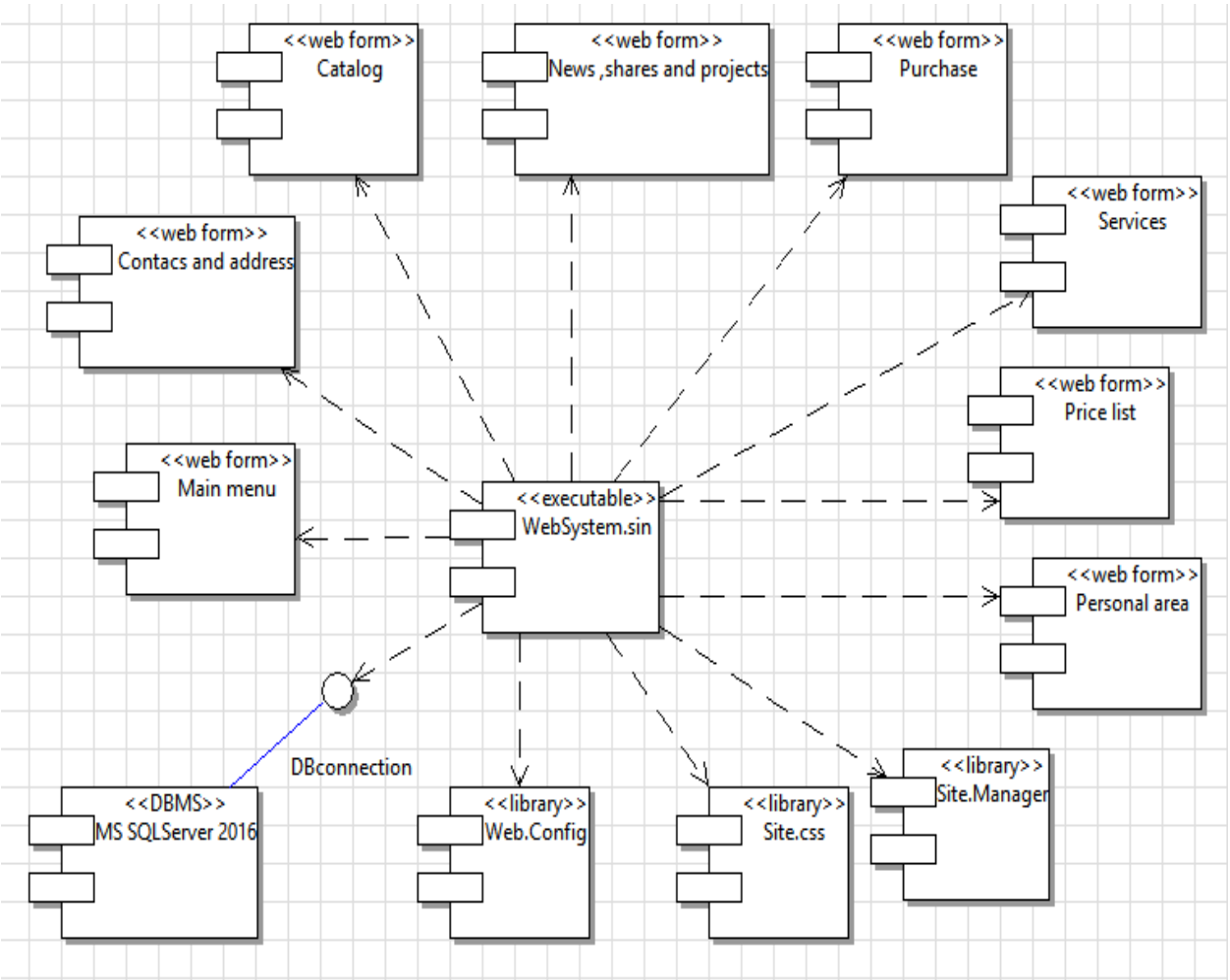



Рисунок 17 – Диаграмма компонентов

Клиентское приложение основано на сочетании форм, соответствующих классов, стереотипов, представленных «формой» и «шрифтом», и библиотек подключаемых модулей. Серверная часть в широком смысле представлена как интеграция между СУБД, которой является MS SQL Server, и самой БД DBLibrary. Последнее основано на наборе связанных таблиц.

## 2.4 Программная реализация мобильного приложения для доступа к веб-сайту компании

Для начала воспользуемся классификацией FURPS2+ для определения требований к нашему мобильному приложению (рисунок 18).

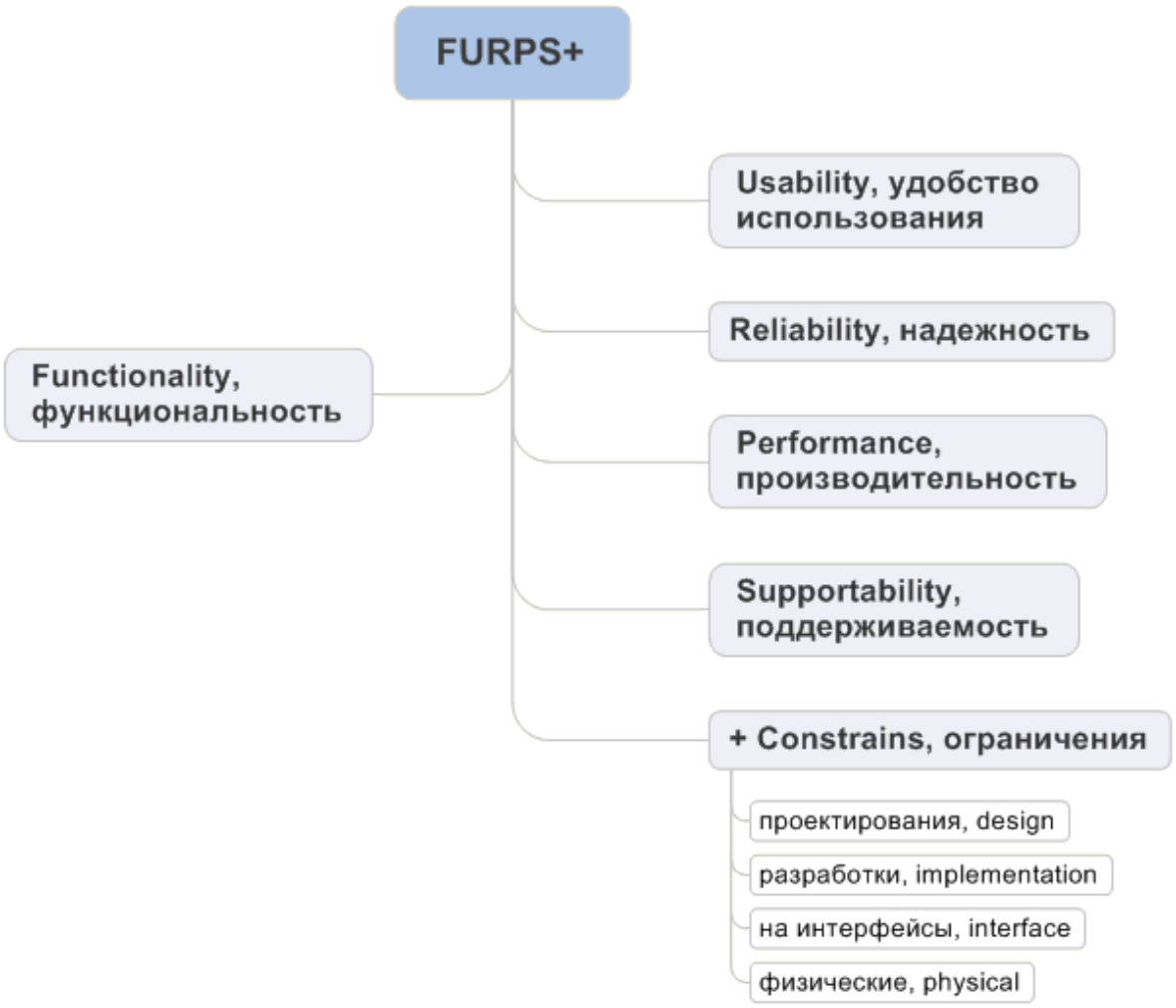



Рисунок 18 – Классификация требований FURPS+

Сформулируем и перечислим требования для нашего приложения, которые должны быть учтены при разработке:

– Functionality: передача данных по запросу онлайн консультации;
– Usability: удобный и понятный интерфейс, комфортная цветовая гамма приложения;
– Reliability: доступность системы всегда;
– Performance: должна запускаться в течении 3 секунд или менее, а время реакции на пользовательские действия не должно превышать 1 секунду;
– Supportability: поддержка операционной системы iOS и Android, корректная работа функционала на различных устройствах [14].

Эта архитектура позволит нам разрабатывать надежные и эффективные мобильные приложения, отвечающие всем функциональным требованиям.

Диаграммы компонентов, показывающие взаимосвязи, зависимости и интерфейсы компонентов программного обеспечения, показаны на рисунках 19 и 20.

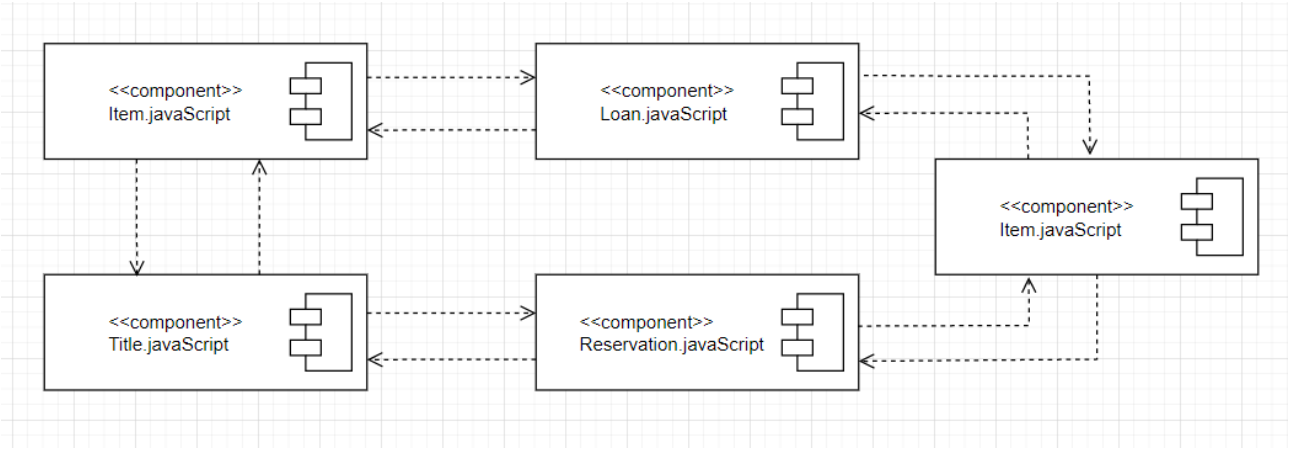



Рисунок 19 – Компонентная модель кода для JavaScript

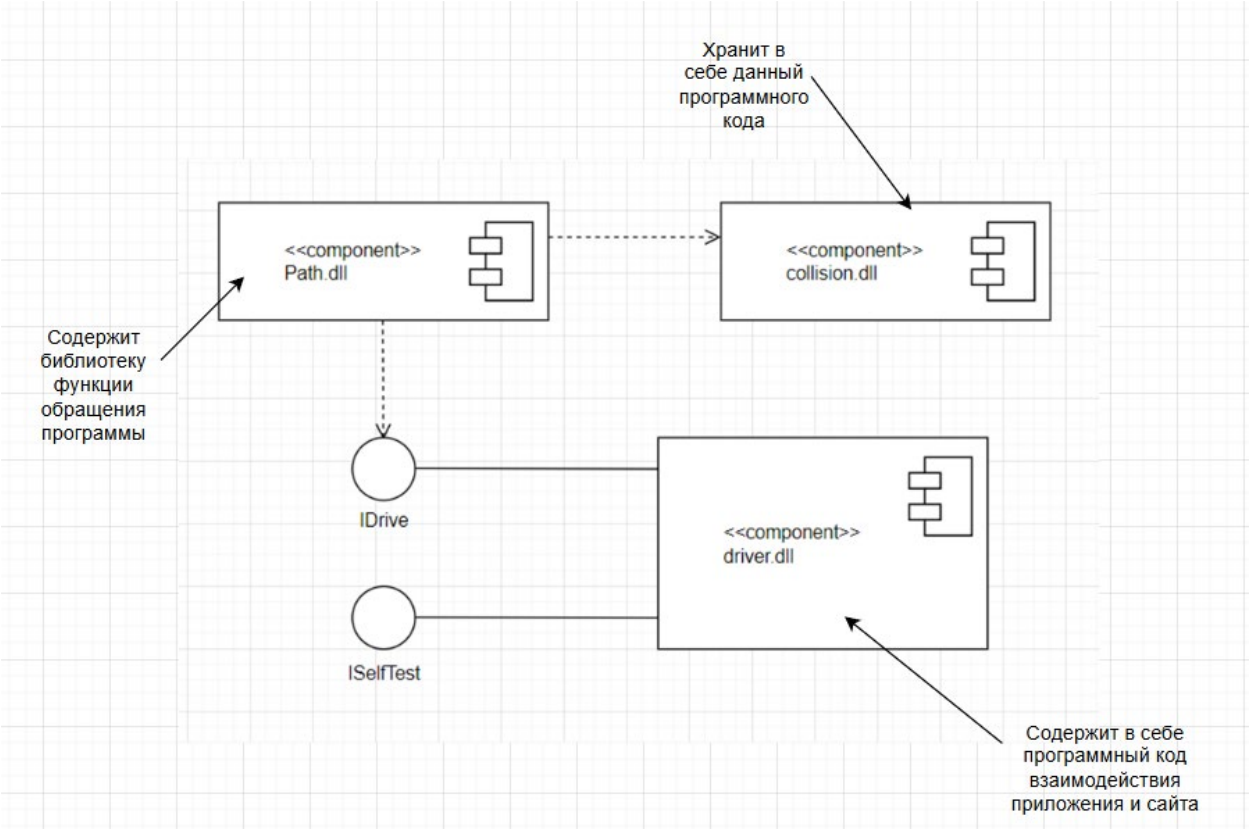



Рисунок 20 – Компонентная модель исполняемой версии

Физическая модель данных – это заключительный этап проектирования базы данных для программного продукта. Разработанная архитектура БД показана на рисунке 21.

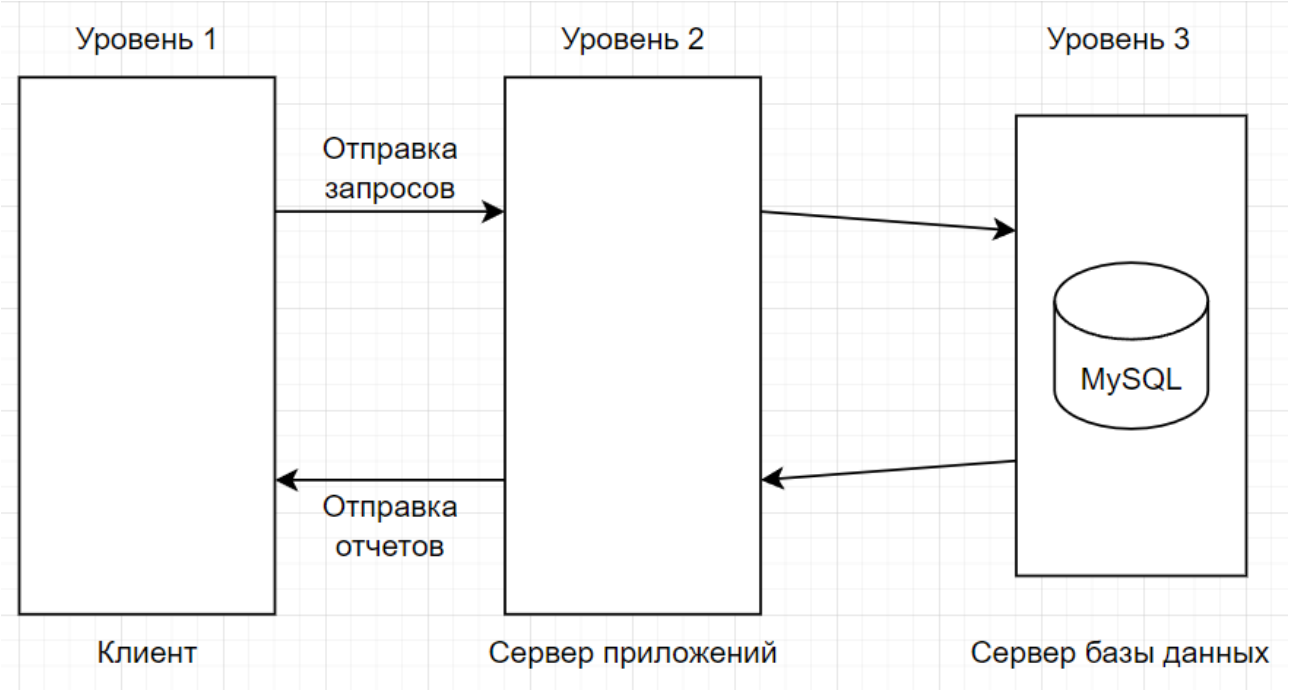
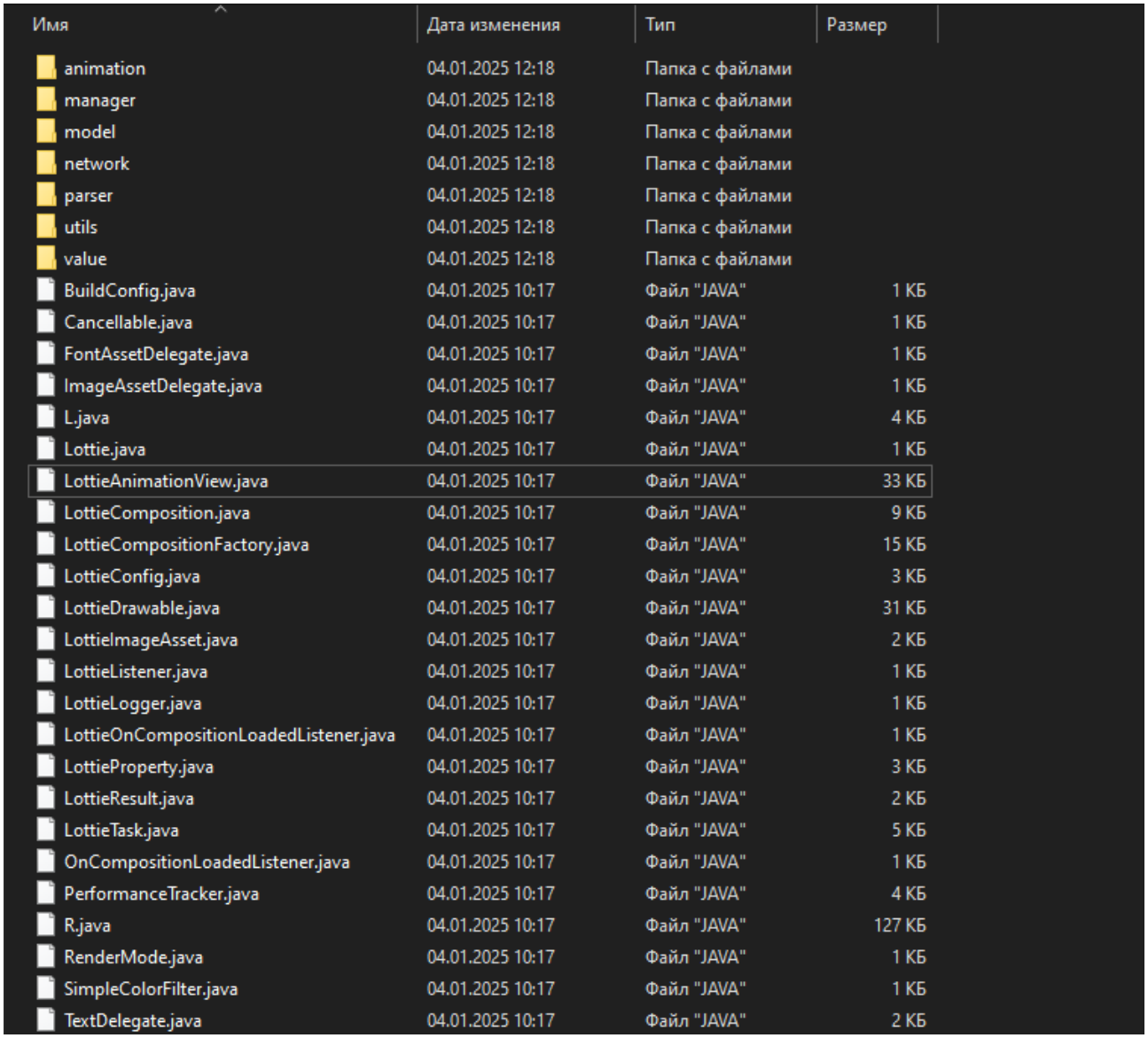



Рисунок 21 – Архитектура СУБД MySQL

После проведения предпроектной реализации, и самой реализации мобильного приложения для доступа к web-сайту компании, была полностью реализовано приложение.

Состав программных файлов можно увидеть на рисунке 22.



| Имя | Дата изменения | Тип | Размер |
|---|---|---|---|
| animation | 04.01.2025 12:18 | Папка с файлами | |
| manager | 04.01.2025 12:18 | Папка с файлами | |
| model | 04.01.2025 12:18 | Папка с файлами | |
| network | 04.01.2025 12:18 | Папка с файлами | |
| parser | 04.01.2025 12:18 | Папка с файлами | |
| utils | 04.01.2025 12:18 | Папка с файлами | |
| value | 04.01.2025 12:18 | Папка с файлами | |
| BuildConfig.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| Cancellable.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| FontAssetDelegate.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| ImageAssetDelegate.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| L.java | 04.01.2025 10:17 | Файл "JAVA" | 4 КБ |
| Lottie.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| LottieAnimationView.java | 04.01.2025 10:17 | Файл "JAVA" | 33 КБ |
| LottieComposition.java | 04.01.2025 10:17 | Файл "JAVA" | 9 КБ |
| LottieCompositionFactory.java | 04.01.2025 10:17 | Файл "JAVA" | 15 КБ |
| LottieConfig.java | 04.01.2025 10:17 | Файл "JAVA" | 3 КБ |
| LottieDrawable.java | 04.01.2025 10:17 | Файл "JAVA" | 31 КБ |
| LottieImageAsset.java | 04.01.2025 10:17 | Файл "JAVA" | 2 КБ |
| LottieListener.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| LottieLogger.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| LottieOnCompositionLoadedListener.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| LottieProperty.java | 04.01.2025 10:17 | Файл "JAVA" | 3 КБ |
| LottieResult.java | 04.01.2025 10:17 | Файл "JAVA" | 2 КБ |
| LottieTask.java | 04.01.2025 10:17 | Файл "JAVA" | 5 КБ |
| OnCompositionLoadedListener.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| PerformanceTracker.java | 04.01.2025 10:17 | Файл "JAVA" | 4 КБ |
| R.java | 04.01.2025 10:17 | Файл "JAVA" | 127 КБ |
| RenderMode.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| SimpleColorFilter.java | 04.01.2025 10:17 | Файл "JAVA" | 1 КБ |
| TextDelegate.java | 04.01.2025 10:17 | Файл "JAVA" | 2 КБ |

Рисунок 22 – Часть программных файлов реализованного мобильного приложения для доступа к web-сайту компании

Программный листинг реализованного мобильного приложения для доступа к web-сайту компании находится в приложении А.

## 2.5 Тестирование запуска мобильного приложения

После процесса программного реализации, проводится тестирование (рисунки 23-25).

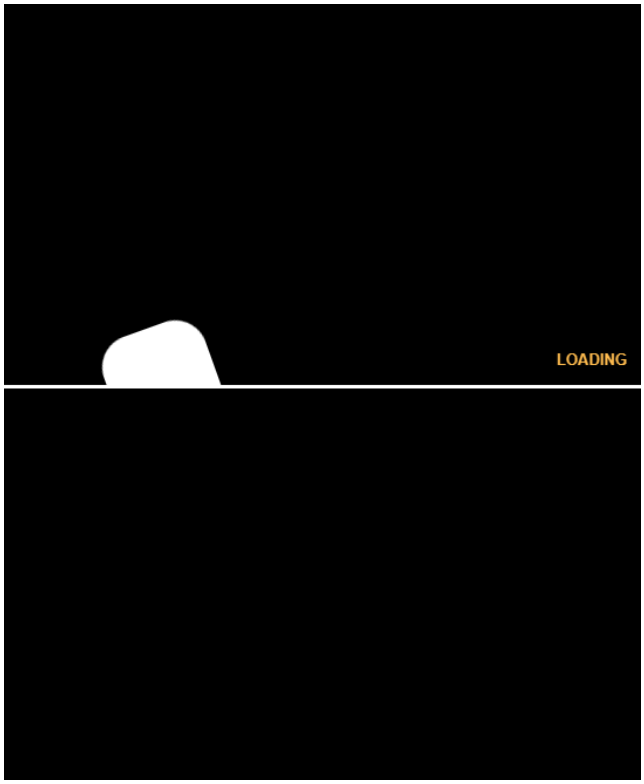



Рисунок 23 – Процесс запуска мобильного приложения


Рисунок 24 – Главная страница сайта компании


Рисунок 25 – Боковое меню сайта компании

Проведем тестирования запуска одного из разделов бокового меню сайта компании (рисунок 26).


Рисунок 26 – Информация о компании

В ходе выполнения практической части проекта было разработано мобильное приложение для доступа к сайту компании «Гильдия мастерских», которое упростило взаимодействие клиентов с сайтом и оптимизировало работу компании.

Были выполнены задачи проведения моделирования, приняты решения по выбору системной архитектуры проекта, проведена разработка и тестирование версии приложения.

Результаты тестирования соответствуют критериям качества продукта.

Наличие собственного мобильного приложения с удобной регистрацией и коммуникацией открывает компании возможности для дальнейшего развития функционала приложения.

## 3 Расчет экономической эффективности проекта

Для расчета срока окупаемости составим список затрат на реализацию проекта, исходим с того, что компании, действующей с существующей стандартной на сегодняшний день ИТ-инфраструктурой, нет необходимости закупать серверное оборудование.

В проекте задействовано 4 сотрудника.

Список затрат:

– серверное оборудование и лицензии на серверное ПО: виртуальный сервер в существующем кластере на базе Ubuntu 20.04 LTS 64-bit= 0 рублей за лицензию;

– IDE для разработки PyCharm: $29.88 в месяц (курс на 11.06.24 82.70 за USD = 2471 рублей в месяц);

– сертификат/домен для сайта: 2000 рублей в год;

– лицензии на СУБД: создание базы данных на существующем сервере СУБД PostgreSQL: 0 рублей;

– система контроля версий GIT: бесплатно;

– зарплаты сотрудников, участвующих в проекте разработки: 320 000 рублей за 2 месяца разработки.

Итоговая сумма первоначальных затрат: 326 942 рублей.

$$Т_{ок} = \frac{К_{п}}{\Delta С} \quad (1)$$

где Кп - капитальные затраты на проект;

ΔС - абсолютное снижение стоимости, оно равняется 16 000 рублей.

$$Т_{ок} = \frac{К_{п}}{\Delta С} = \frac{326942}{16000} = 20 \text{ (месяцев)}$$

На основе проведенного экономического анализа выявлено, что общие расходы на проектирование и разработку мобильного приложения составляют 326 942 рублей. Кроме того, был проведен расчет срока окупаемости, который составил 20 месяцев.

Таким образом, с точки зрения экономического анализа, приложение может считаться экономически выгодным, так как затраты окупятся за короткий промежуток времени.

# Заключение

В процессе написания работы, была мною достигнута установленная цель, которая заключается в теоретическом и практическом анализе по разработке мобильного приложения для доступа к веб-сайту для компании ООО «Гильдия мастерских».

Для достижения поставленной цели, решили следующие задачи:

- изучили организационную структуру предприятия с распределением ответственности, полномочий и взаимоотношений между работниками предприятия;
- построили функциональную модель и/или процессную модель организации «как есть», «как должно быть» в любых нотациях (IDEF, UML, ARIS, BPMN);
- описали и обосновали решения по системной архитектуре проекта;
- разработали мобильное приложение для доступа к веб-сайту;
- провели расчет экономической эффективности проекта.

Программная реализация прошла успешно, приложение работает исправно.

Результатом выпускной квалификационной работы стало мобильное приложение для организации доступа к сайту, которое обеспечило пользователей удобным средством взаимодействия и коммуникации с сайтом, и оптимизировало работу компании.

Компанией рассматривается дальнейшее расширение функциональных возможностей мобильного приложения, автоматизирующее различные бизнес-процессы.

Приложение привлекает новых пользователей сервиса и увеличивает количество довольных клиентов, что способствует повышению экономической эффективности. Разработанное мобильное приложение также положительно влияет на статус компании, создавая образ современной организации с широкими возможностями.

## Список используемой литературы

Основная литература

1. Баланов, А. Н. Комплексное руководство по разработке: от мобильных приложений до веб-технологий : учебное пособие для спо / А. Н. Баланов. — Санкт-Петербург : Лань, 2024. — 64 с. — ISBN 978-5-507-48842-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/394580 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

2. Баланов, А. Н. Цифровые платформы и системы : учебное пособие для вузов / А. Н. Баланов. — Санкт-Петербург : Лань, 2024. — 452 с. — ISBN 978-5-507-49532-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/424577 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

3. Ермаков, С. Р. Основы веб-разработки : учебное пособие / С. Р. Ермаков, П. В. Беляев, А. В. Симонова. — Москва : РТУ МИРЭА, 2024. — 181 с. — ISBN 978-5-7339-2147-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/420965 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

4. Зорина, Н. В. Программирование на языке Джава : учебное пособие / Н. В. Зорина. — Москва : РТУ МИРЭА, 2023 — Часть 2 — 2023. — 82 с. — ISBN 978-5-7339-1765-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/368972 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

5. Иванова, Е. А. Кроссплатформенные приложения : учебное пособие / Е. А. Иванова, Т. А. Крамаренко. — Краснодар : КубГАУ, 2020. — 165 с. — ISBN 978-5-907346-93-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/254237 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

6. Калгина, И. С. Разработка мобильных приложений : учебное пособие / И. С. Калгина. — Чита : ЗабГУ, 2022. — 163 с. — ISBN 978-5-9293-3137-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/363323 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

7. Крыжановская, Ю. А. Основы JAVA : учебное пособие / Ю. А. Крыжановская, В. Г. Ляликова, М. М. Безрядин. — Воронеж : ВГУ, 2020. — 96 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/433010 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

8. Льюис, Ш. Нативная разработка мобильных приложений : руководство / Ш. Льюис, М. Данн ; перевод с английского А. Н. Киселева. — Москва : ДМК Пресс, 2020. — 376 с. — ISBN 978-5-97060-845-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/179491 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

9. Молчанова, Е. И. Объектно-ориентированное программирование. Основы объектного программирования на языке Java в среде IDE NetBeans : учебное пособие : в 2 частях / Е. И. Молчанова, В. В. Федоров. — Иркутск : ИрГУПС, 2022 — Часть 1 — 2022. — 128 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/342125 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

10. Неволин, А. О. Архитектура вычислительных устройств и их программирование : учебное пособие / А. О. Неволин. — Москва : Горячая линия-Телеком, 2020. — 80 с. — ISBN 978-5-9912-0878-9. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/267848 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

11. Пономарчук, Ю. В. Программирование на языке Java : учебное пособие / Ю. В. Пономарчук, И. В. Кузнецов. — Хабаровск : ДВГУПС, 2021.

— 103 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/259451 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

12. Разработка графического интерфейса пользователя информационной системы с использованием библиотеки Qt : учебное пособие / Ю. В. Минин, А. И. Елисеев, В. В. Алексеев, Ю. А. Губсков. — Тамбов : ТГТУ, 2021. — 84 с. — ISBN 978-5-8265-2397-1. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/320516 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

13. Разработка приложений для ОС Android : учебное пособие / И. В. Кузнецов, М. С. Исаев, Ю. В. Пономарчук, А. А. Холодилов. — Хабаровск : ДВГУПС, 2023. — 112 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/433625 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

14. Федоричев, Л. А. Реализация многопоточности в языке Java / Л. А. Федоричев, О. В. Букунова. — Санкт-Петербург : Лань, 2024. — 72 с. — ISBN 978-5-507-48153-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/367400 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

15. Хабитуев, Б. В. Программирование на языке Java: практикум : учебное пособие / Б. В. Хабитуев. — Улан-Удэ : БГУ, 2020. — 94 с. — ISBN 978-5-9793-1548-5. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/171791 (дата обращения: 16.01.2025). — Режим доступа: для авториз. пользователей.

Электронные источники

16. Блог на хабре о разработке под Android [Электронный ресурс]. – Режим доступа: www.habrahabr.ru

17. Официальная справка для Android разработчиков [Электронный ресурс]. – Режим доступа: https://developer.android.com.

18. Программирование для Android [Электронный ресурс]. – Режим доступа: http://flashbot.ru/android-dev

19. Официальная справка по среде программирования [Электронный ресурс]. – Режим доступа: http://www.jetbrains.com

20. Форум о программировании для Android [Электронный ресурс]. – Режим доступа: http://www.cyberforum.ru/android-dev/

21. Java Development Kit. Википедия [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Java_Development_Kit

22. Что такое Android SDK – коротко и информативно SDK [Электронный ресурс]. – Режим доступа: ipodman.ru.

23. Проектирование информационных систем [Электронный ресурс]. – Режим доступа: www.intuit.ru

24. Get the Android SDK. [Электронный ресурс]. – Режим доступа: http://developer.android.com/sdk/index.htm

25. Компоненты приложений в Android. Часть 1. [Электронный ресурс]. – Режим доступа: http://android-shark.ru/komponentyi-prilozheniy-v-android-chast

26. Компоненты приложений в Android. Часть 2. [Электронный ресурс]. – Режим доступа: http://eclipse-3.narod.ru/#_Toc90528393

27. Официальный сайт Android. [Электронный ресурс]. – Режим доступа: http://www.android.com

28. Что это за Android? [Электронный ресурс]. – Режим доступа: http://android.com.ua/android_os.html

29. Обобщенный Model-View-Controller [Электронный ресурс]. – Режим доступа: http://www.rsdn.ru/article/patterns/generic-mvc.xml

30. JavaServer Pages Technology [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/javaee/jsp/index.html

31. Unified Expression Language [Электронный ресурс]. – Режим доступа: http://www.oracle.com/technetwork/java/unifiedel-139263.html

32. Introducing JSON [Электронный ресурс]. – Режим доступа: http://www.json.org

33. Duvall, P. Matyas, S. Glover, A. Continuous Integration: Improving Software Quality and Reducing Risk. – Addison-Wesley, 2007.

34. Meet Jenkins [Электронный ресурс]. – Режим доступа: https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins

35. Marmalade C++ SDK [Электронный ресурс]. – Режим доступа: https://www.madewithmarmalade.com/products/marmalade-sdk

36. Cocos2d-x [Электронный ресурс]. – Режим доступа: http://www.cocos2d-x.org/products

37. Feature/Platform Support [Электронный ресурс]. – Режим доступа: http://www.mosync.com/docs/sdk/tools/references/feature-platform-support/index

38. RhoMobile Suite Docs [Электронный ресурс]. – Режим доступа: http://docs.rhomobile.com/en/4.1.0/home

39. Documentation [Электронный ресурс]. – Режим доступа: http://docs.appcelerator.com/titanium/latest/#!/guide/Quick_Start

40. Guides [Электронный ресурс]. – Режим доступа: http://docs.phonegap.com/en/3.4.0/index.html

Приложение А

## Программный листинг мобильного приложения

```
package androidx.activity;

import android.app.Activity;
import android.content.Context;
import android.content.Intent;
import android.content.IntentSender;
import android.content.res.Configuration;
import android.os.Build;
import android.os.Bundle;
import android.os.Handler;
import android.os.Looper;
import android.os.SystemClock;
import android.text.TextUtils;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewGroup;
import android.view.ViewTreeObserver;
import android.view.Window;
import android.window.OnBackInvokedDispatcher;
import androidx.activity.contextaware.ContextAware;
import androidx.activity.contextaware.ContextAwareHelper;
import
androidx.activity.contextaware.OnContextAvailableListener;
import androidx.activity.result.ActivityResultCallback;
import androidx.activity.result.ActivityResultCaller;
import androidx.activity.result.ActivityResultLauncher;
import androidx.activity.result.ActivityResultRegistry;
import androidx.activity.result.ActivityResultRegistryOwner;
import androidx.activity.result.IntentSenderRequest;
import androidx.activity.result.contract.ActivityResultContract;
import
androidx.activity.result.contract.ActivityResultContracts;
import androidx.core.app.ActivityCompat;
import androidx.core.app.ActivityOptionsCompat;
import androidx.core.app.MultiWindowModeChangedInfo;
import androidx.core.app.OnMultiWindowModeChangedProvider;
import androidx.core.app.OnNewIntentProvider;
import androidx.core.app.OnPictureInPictureModeChangedProvider;
import androidx.core.app.PictureInPictureModeChangedInfo;
import androidx.core.content.ContextCompat;
import androidx.core.content.OnConfigurationChangedProvider;
import androidx.core.content.OnTrimMemoryProvider;
import androidx.core.os.BuildCompat;
import androidx.core.util.Consumer;
import androidx.core.view.MenuHost;
import androidx.core.view.MenuHostHelper;
```

```
import androidx.core.view.MenuProvider;
import androidx.lifecycle.HasDefaultViewModelProviderFactory;
import androidx.lifecycle.Lifecycle;
import androidx.lifecycle.LifecycleEventObserver;
import androidx.lifecycle.LifecycleOwner;
import androidx.lifecycle.LifecycleRegistry;
import androidx.lifecycle.ReportFragment;
import androidx.lifecycle.SavedStateHandleSupport;
import androidx.lifecycle.SavedStateViewModelFactory;
import androidx.lifecycle.ViewModelProvider;
import androidx.lifecycle.ViewModelStore;
import androidx.lifecycle.ViewModelStoreOwner;
import androidx.lifecycle.ViewTreeLifecycleOwner;
import androidx.lifecycle.ViewTreeViewModelStoreOwner;
import androidx.lifecycle.viewmodel.CreationExtras;
import androidx.lifecycle.viewmodel.MutableCreationExtras;
import androidx.savedstate.SavedStateRegistry;
import androidx.savedstate.SavedStateRegistryController;
import androidx.savedstate.SavedStateRegistryOwner;
import androidx.savedstate.ViewTreeSavedStateRegistryOwner;
import androidx.tracing.Trace;
import androidx.work.WorkRequest;
import java.util.Iterator;
import java.util.concurrent.CopyOnWriteArrayList;
import java.util.concurrent.Executor;
import java.util.concurrent.atomic.AtomicInteger;
import kotlin.Unit;

public class ComponentActivity extends
androidx.core.app.ComponentActivity implements ContextAware,
LifecycleOwner, ViewModelStoreOwner,
HasDefaultViewModelProviderFactory, SavedStateRegistryOwner,
OnBackPressedDispatcherOwner, ActivityResultRegistryOwner,
ActivityResultCaller, OnConfigurationChangedProvider,
OnTrimMemoryProvider, OnNewIntentProvider,
OnMultiWindowModeChangedProvider,
OnPictureInPictureModeChangedProvider, MenuHost,
FullyDrawnReporterOwner {
      private static final String ACTIVITY_RESULT_TAG =
"android:support:activity-result";
            private final ActivityResultRegistry
mActivityResultRegistry;
    private int mContentLayoutId;
    final ContextAwareHelper mContextAwareHelper;
    private ViewModelProvider.Factory mDefaultFactory;
    private boolean mDispatchingOnMultiWindowModeChanged;
    private boolean mDispatchingOnPictureInPictureModeChanged;
    final FullyDrawnReporter mFullyDrawnReporter;
    private final LifecycleRegistry mLifecycleRegistry;
    private final MenuHostHelper mMenuHostHelper;
```

```
private final AtomicInteger mNextLocalRequestCode;
            private         final         OnBackPressedDispatcher
mOnBackPressedDispatcher;
     private  final  CopyOnWriteArrayList<Consumer<Configuration>>
mOnConfigurationChangedListeners;
                              private                              final
CopyOnWriteArrayList<Consumer<MultiWindowModeChangedInfo>>
mOnMultiWindowModeChangedListeners;
       private    final    CopyOnWriteArrayList<Consumer<Intent>>
mOnNewIntentListeners;
                              private                              final
CopyOnWriteArrayList<Consumer<PictureInPictureModeChangedInfo>>
mOnPictureInPictureModeChangedListeners;
       private    final    CopyOnWriteArrayList<Consumer<Integer>>
mOnTrimMemoryListeners;
    final ReportFullyDrawnExecutor mReportFullyDrawnExecutor;
                  final               SavedStateRegistryController
mSavedStateRegistryController;
    private ViewModelStore mViewModelStore;

     private  interface  ReportFullyDrawnExecutor  extends  Executor
{
        void activityDestroyed();

        void viewCreated(View view);
    }

    @Deprecated
    public Object onRetainCustomNonConfigurationInstance() {
        return null;
    }

    static final class NonConfigurationInstances {
        Object custom;
        ViewModelStore viewModelStore;

        NonConfigurationInstances() {
        }
    }

    /* access modifiers changed from: package-private */
       /*    renamed    from:    lambda$new$0$androidx-activity-
ComponentActivity  reason: not valid java name */
          public        /*        synthetic        */        Unit
m0lambda$new$0$androidxactivityComponentActivity() {
        reportFullyDrawn();
        return null;
    }

    public ComponentActivity() {
```

```
 this.mContextAwareHelper = new ContextAwareHelper();
              this.mMenuHostHelper   =   new   MenuHostHelper(new
ComponentActivity$$ExternalSyntheticLambda0(this));
        this.mLifecycleRegistry = new LifecycleRegistry(this);
                     SavedStateRegistryController     create     =
SavedStateRegistryController.create(this);
        this.mSavedStateRegistryController = create;
                      this.mOnBackPressedDispatcher     =     new
OnBackPressedDispatcher(new Runnable() {
            public void run() {
                try {
                    ComponentActivity.super.onBackPressed();
                } catch (IllegalStateException e) {
                      if (!TextUtils.equals(e.getMessage(), "Can
not perform this action after onSaveInstanceState")) {
                        throw e;
                    }
                }
            }
        });
            ReportFullyDrawnExecutor  createFullyDrawnExecutor  =
createFullyDrawnExecutor();
                           this.mReportFullyDrawnExecutor       =
createFullyDrawnExecutor;
                        this.mFullyDrawnReporter      =      new
FullyDrawnReporter(createFullyDrawnExecutor,                new
ComponentActivity$$ExternalSyntheticLambda1(this));
        this.mNextLocalRequestCode = new AtomicInteger();
                      this.mActivityResultRegistry     =     new
ActivityResultRegistry() {
                  public  <I,  O>  void  onLaunch(final  int  i,
ActivityResultContract<I,   O>   activityResultContract,   I   i2,
ActivityOptionsCompat activityOptionsCompat) {
                         ComponentActivity  componentActivity  =
ComponentActivity.this;
                                                              final
ActivityResultContract.SynchronousResult<O>  synchronousResult  =
activityResultContract.getSynchronousResult(componentActivity,
i2);
                if (synchronousResult != null) {
                     new Handler(Looper.getMainLooper()).post(new
Runnable() {
                        public void run() {
                             AnonymousClass2.this.dispatchResult(
i, synchronousResult.getValue());
                        }
                    });
                    return;
                }
```

```
                                    Intent      createIntent      =
activityResultContract.createIntent(componentActivity, i2);
                Bundle bundle = null;
                      if (createIntent.getExtras() != null &&
createIntent.getExtras().getClassLoader() == null) {
                    createIntent.setExtrasClassLoader(componentA
ctivity.getClassLoader());
                }
                                                                  if
(createIntent.hasExtra(ActivityResultContracts.StartActivityForR
esult.EXTRA_ACTIVITY_OPTIONS_BUNDLE)) {
                                                        bundle    =
createIntent.getBundleExtra(ActivityResultContracts.StartActivit
yForResult.EXTRA_ACTIVITY_OPTIONS_BUNDLE);
                    createIntent.removeExtra(ActivityResultContr
acts.StartActivityForResult.EXTRA_ACTIVITY_OPTIONS_BUNDLE);
                } else if (activityOptionsCompat != null) {
                    bundle = activityOptionsCompat.toBundle();
                }
                Bundle bundle2 = bundle;
                                                                  if
(ActivityResultContracts.RequestMultiplePermissions.ACTION_REQUE
ST_PERMISSIONS.equals(createIntent.getAction())) {
                                    String[]  stringArrayExtra  =
createIntent.getStringArrayExtra(ActivityResultContracts.Request
MultiplePermissions.EXTRA_PERMISSIONS);
                    if (stringArrayExtra == null) {
                        stringArrayExtra = new String[0];
                    }
                    ActivityCompat.requestPermissions(componentA
ctivity, stringArrayExtra, i);
                                                    }    else    if
(ActivityResultContracts.StartIntentSenderForResult.ACTION_INTEN
T_SENDER_REQUEST.equals(createIntent.getAction())) {
                        IntentSenderRequest intentSenderRequest =
(IntentSenderRequest)
createIntent.getParcelableExtra(ActivityResultContracts.StartInt
entSenderForResult.EXTRA_INTENT_SENDER_REQUEST);
                    try {
                        ActivityCompat.startIntentSenderForResul
t(componentActivity,  intentSenderRequest.getIntentSender(),  i,
intentSenderRequest.getFillInIntent(),
intentSenderRequest.getFlagsMask(),
intentSenderRequest.getFlagsValues(), 0, bundle2);
                    } catch (IntentSender.SendIntentException e)
{
                                                                  new
Handler(Looper.getMainLooper()).post(new Runnable() {
                            public void run() {
```

```
  AnonymousClass2.this.dispatchResult(i,          0,          new
Intent().setAction(ActivityResultContracts.StartIntentSenderForR
esult.ACTION_INTENT_SENDER_REQUEST).putExtra(ActivityResultContr
acts.StartIntentSenderForResult.EXTRA_SEND_INTENT_EXCEPTION,
e));
                                }
                            });
                        }
                    } else {
                         ActivityCompat.startActivityForResult(compon
entActivity, createIntent, i, bundle2);
                    }
                }
            };
                    this.mOnConfigurationChangedListeners    =    new
CopyOnWriteArrayList<>();
                          this.mOnTrimMemoryListeners      =      new
CopyOnWriteArrayList<>();
                           this.mOnNewIntentListeners      =      new
CopyOnWriteArrayList<>();
                  this.mOnMultiWindowModeChangedListeners   =   new
CopyOnWriteArrayList<>();
              this.mOnPictureInPictureModeChangedListeners  =  new
CopyOnWriteArrayList<>();
            this.mDispatchingOnMultiWindowModeChanged = false;
            this.mDispatchingOnPictureInPictureModeChanged = false;
            if (getLifecycle() != null) {
                if (Build.VERSION.SDK_INT >= 19) {
                                      getLifecycle().addObserver(new
LifecycleEventObserver() {
                           public void onStateChanged(LifecycleOwner
lifecycleOwner, Lifecycle.Event event) {
                            if (event == Lifecycle.Event.ON_STOP) {
                                                Window   window   =
ComponentActivity.this.getWindow();
                                  View peekDecorView = window != null
? window.peekDecorView() : null;
                                if (peekDecorView != null) {
                                     Api19Impl.cancelPendingInputEven
ts(peekDecorView);
                                }
                            }
                        }
                    });
                }
                                      getLifecycle().addObserver(new
LifecycleEventObserver() {
                         public  void  onStateChanged(LifecycleOwner
lifecycleOwner, Lifecycle.Event event) {
                       if (event == Lifecycle.Event.ON_DESTROY) {
```

```
    ComponentActivity.this.mContextAwareHelper.clearAvailableCont
ext();
                                                                    if
(!ComponentActivity.this.isChangingConfigurations()) {
                                ComponentActivity.this.getViewModelS
tore().clear();
                            }
                             ComponentActivity.this.mReportFullyDrawn
Executor.activityDestroyed();
                        }
                    }
                });
                                        getLifecycle().addObserver(new
LifecycleEventObserver() {
                         public  void  onStateChanged(LifecycleOwner
lifecycleOwner, Lifecycle.Event event) {
                        ComponentActivity.this.ensureViewModelStore(
);
                        ComponentActivity.this.getLifecycle().remove
Observer(this);
                    }
                });
                create.performAttach();
                 SavedStateHandleSupport.enableSavedStateHandles(this
);
                          if    (19    <=    Build.VERSION.SDK_INT    &&
Build.VERSION.SDK_INT <= 23) {
                                        getLifecycle().addObserver(new
ImmLeaksCleaner(this));
                }
                 getSavedStateRegistry().registerSavedStateProvider(A
CTIVITY_RESULT_TAG,                                                 new
ComponentActivity$$ExternalSyntheticLambda2(this));
                                    addOnContextAvailableListener(new
ComponentActivity$$ExternalSyntheticLambda3(this));
                return;
            }
             throw new IllegalStateException("getLifecycle() returned
null in ComponentActivity's constructor. Please make sure you
are lazily constructing your Lifecycle in the first call to
getLifecycle() rather than relying on field initialization.");
        }

        /* access modifiers changed from: package-private */
           /*    renamed    from:    lambda$new$1$androidx-activity-
ComponentActivity  reason: not valid java name */
              public       /*       synthetic       */       Bundle
m1lambda$new$1$androidxactivityComponentActivity() {
            Bundle bundle = new Bundle();
```

```
  this.mActivityResultRegistry.onSaveInstanceState(bundle);
        return bundle;
    }

    /* access modifiers changed from: package-private */
       /*    renamed    from:    lambda$new$2$androidx-activity-
ComponentActivity  reason: not valid java name */
          public       /*       synthetic       */       void
m2lambda$new$2$androidxactivityComponentActivity(Context
context) {
                     Bundle     consumeRestoredStateForKey     =
getSavedStateRegistry().consumeRestoredStateForKey(ACTIVITY_RESU
LT_TAG);
        if (consumeRestoredStateForKey != null) {
             this.mActivityResultRegistry.onRestoreInstanceState(
consumeRestoredStateForKey);
        }
    }

    public ComponentActivity(int i) {
        this();
        this.mContentLayoutId = i;
    }

    /* access modifiers changed from: protected */
    public void onCreate(Bundle bundle) {
         this.mSavedStateRegistryController.performRestore(bundle
);
         this.mContextAwareHelper.dispatchOnContextAvailable(this
);
        super.onCreate(bundle);
        ReportFragment.injectIfNeededIn(this);
        if (BuildCompat.isAtLeastT()) {
             this.mOnBackPressedDispatcher.setOnBackInvokedDispat
cher(Api33Impl.getOnBackInvokedDispatcher(this));
        }
        int i = this.mContentLayoutId;
        if (i != 0) {
            setContentView(i);
        }
    }

    /* access modifiers changed from: protected */
    public void onSaveInstanceState(Bundle bundle) {
        Lifecycle lifecycle = getLifecycle();
        if (lifecycle instanceof LifecycleRegistry) {
                                          ((LifecycleRegistry)
lifecycle).setCurrentState(Lifecycle.State.CREATED);
        }
        super.onSaveInstanceState(bundle);
```

```
  this.mSavedStateRegistryController.performSave(bundle);
    }

    public final Object onRetainNonConfigurationInstance() {
        NonConfigurationInstances nonConfigurationInstances;
              Object   onRetainCustomNonConfigurationInstance   =
onRetainCustomNonConfigurationInstance();
        ViewModelStore viewModelStore = this.mViewModelStore;
         if (viewModelStore == null && (nonConfigurationInstances
=  (NonConfigurationInstances)  getLastNonConfigurationInstance())
!= null) {
                                                viewModelStore       =
nonConfigurationInstances.viewModelStore;
        }
                     if     (viewModelStore     ==     null     &&
onRetainCustomNonConfigurationInstance == null) {
            return null;
        }
          NonConfigurationInstances  nonConfigurationInstances2  =
new NonConfigurationInstances();
                          nonConfigurationInstances2.custom      =
onRetainCustomNonConfigurationInstance;
                   nonConfigurationInstances2.viewModelStore    =
viewModelStore;
        return nonConfigurationInstances2;
    }

    @Deprecated
    public Object getLastCustomNonConfigurationInstance() {
          NonConfigurationInstances  nonConfigurationInstances  =
(NonConfigurationInstances) getLastNonConfigurationInstance();
        if (nonConfigurationInstances != null) {
            return nonConfigurationInstances.custom;
        }
        return null;
    }

    public void setContentView(int i) {
        initViewTreeOwners();
         this.mReportFullyDrawnExecutor.viewCreated(getWindow().g
etDecorView());
        super.setContentView(i);
    }

    public void setContentView(View view) {
        initViewTreeOwners();
         this.mReportFullyDrawnExecutor.viewCreated(getWindow().g
etDecorView());
        super.setContentView(view);
   }
```

```
 public void setContentView(View view, ViewGroup.LayoutParams
layoutParams) {
        initViewTreeOwners();
         this.mReportFullyDrawnExecutor.viewCreated(getWindow().g
etDecorView());
        super.setContentView(view, layoutParams);
    }

    public void addContentView(View view, ViewGroup.LayoutParams
layoutParams) {
        initViewTreeOwners();
         this.mReportFullyDrawnExecutor.viewCreated(getWindow().g
etDecorView());
        super.addContentView(view, layoutParams);
    }

    private void initViewTreeOwners() {
           ViewTreeLifecycleOwner.set(getWindow().getDecorView(),
this);
         ViewTreeViewModelStoreOwner.set(getWindow().getDecorView
(), this);
         ViewTreeSavedStateRegistryOwner.set(getWindow().getDecor
View(), this);
         ViewTreeOnBackPressedDispatcherOwner.set(getWindow().get
DecorView(), this);
         ViewTreeFullyDrawnReporterOwner.set(getWindow().getDecor
View(), this);
    }

    public Context peekAvailableContext() {
        return this.mContextAwareHelper.peekAvailableContext();
    }

                  public                final                void
addOnContextAvailableListener(OnContextAvailableListener
onContextAvailableListener) {
         this.mContextAwareHelper.addOnContextAvailableListener(o
nContextAvailableListener);
    }

                  public                final                void
removeOnContextAvailableListener(OnContextAvailableListener
onContextAvailableListener) {
         this.mContextAwareHelper.removeOnContextAvailableListene
r(onContextAvailableListener);
    }

    public boolean onPreparePanel(int i, View view, Menu menu) {
        if (i != 0) {
            return true;
```

```
    }
        super.onPreparePanel(i, view, menu);
        this.mMenuHostHelper.onPrepareMenu(menu);
        return true;
    }

    public boolean onCreatePanelMenu(int i, Menu menu) {
        if (i != 0) {
            return true;
        }
        super.onCreatePanelMenu(i, menu);
                        this.mMenuHostHelper.onCreateMenu(menu,
getMenuInflater());
        return true;
    }

    public boolean onMenuItemSelected(int i, MenuItem menuItem)
{
        if (super.onMenuItemSelected(i, menuItem)) {
            return true;
        }
        if (i == 0) {
                                                          return
this.mMenuHostHelper.onMenuItemSelected(menuItem);
        }
        return false;
    }

    public void onPanelClosed(int i, Menu menu) {
        this.mMenuHostHelper.onMenuClosed(menu);
        super.onPanelClosed(i, menu);
    }

    public void addMenuProvider(MenuProvider menuProvider) {
        this.mMenuHostHelper.addMenuProvider(menuProvider);
    }

      public   void   addMenuProvider(MenuProvider   menuProvider,
LifecycleOwner lifecycleOwner) {
                this.mMenuHostHelper.addMenuProvider(menuProvider,
lifecycleOwner);
    }

      public   void   addMenuProvider(MenuProvider   menuProvider,
LifecycleOwner lifecycleOwner, Lifecycle.State state) {
                this.mMenuHostHelper.addMenuProvider(menuProvider,
lifecycleOwner, state);
    }

    public void removeMenuProvider(MenuProvider menuProvider) {
```

```
    this.mMenuHostHelper.removeMenuProvider(menuProvider);
    }

    public void invalidateMenu() {
        invalidateOptionsMenu();
    }

    public Lifecycle getLifecycle() {
        return this.mLifecycleRegistry;
    }

    public ViewModelStore getViewModelStore() {
        if (getApplication() != null) {
            ensureViewModelStore();
            return this.mViewModelStore;
        }
         throw new IllegalStateException("Your activity is not
yet attached to the Application instance. You can't request
ViewModel before onCreate call.");
    }

    /* access modifiers changed from: package-private */
    public void ensureViewModelStore() {
        if (this.mViewModelStore == null) {
              NonConfigurationInstances nonConfigurationInstances
= (NonConfigurationInstances) getLastNonConfigurationInstance();
            if (nonConfigurationInstances != null) {
                                        this.mViewModelStore    =
nonConfigurationInstances.viewModelStore;
            }
            if (this.mViewModelStore == null) {
                this.mViewModelStore = new ViewModelStore();
            }
        }
    }

                  public                ViewModelProvider.Factory
getDefaultViewModelProviderFactory() {
        if (this.mDefaultFactory == null) {
                                this.mDefaultFactory     =     new
SavedStateViewModelFactory(getApplication(),   this,   getIntent()
!= null ? getIntent().getExtras() : null);
        }
        return this.mDefaultFactory;
    }

    public CreationExtras getDefaultViewModelCreationExtras() {
            MutableCreationExtras  mutableCreationExtras  =  new
MutableCreationExtras();
        if (getApplication() != null) {
```

```
            mutableCreationExtras.set(ViewModelProvider.AndroidView
ModelFactory.APPLICATION_KEY, getApplication());
        }
            mutableCreationExtras.set(SavedStateHandleSupport.SAVED_
STATE_REGISTRY_OWNER_KEY, this);
            mutableCreationExtras.set(SavedStateHandleSupport.VIEW_M
ODEL_STORE_OWNER_KEY, this);
            if (!(getIntent() == null || getIntent().getExtras() ==
null)) {
                mutableCreationExtras.set(SavedStateHandleSupport.DE
FAULT_ARGS_KEY, getIntent().getExtras());
        }
        return mutableCreationExtras;
    }

    public void onBackPressed() {
        this.mOnBackPressedDispatcher.onBackPressed();
    }

              public          final          OnBackPressedDispatcher
getOnBackPressedDispatcher() {
        return this.mOnBackPressedDispatcher;
    }

    public final SavedStateRegistry getSavedStateRegistry() {
                                                              return
this.mSavedStateRegistryController.getSavedStateRegistry();
    }

    public FullyDrawnReporter getFullyDrawnReporter() {
        return this.mFullyDrawnReporter;
    }

    @Deprecated
    public void startActivityForResult(Intent intent, int i) {
        super.startActivityForResult(intent, i);
    }

    @Deprecated
     public  void  startActivityForResult(Intent  intent,  int  i,
Bundle bundle) {
        super.startActivityForResult(intent, i, bundle);
    }

    @Deprecated
       public    void    startIntentSenderForResult(IntentSender
intentSender,  int  i,  Intent  intent,  int  i2,  int  i3,  int  i4)
throws IntentSender.SendIntentException {
              super.startIntentSenderForResult(intentSender,   i,
intent, i2, i3, i4);
```

```
    }

    @Deprecated
       public    void    startIntentSenderForResult(IntentSender
intentSender, int i, Intent intent, int i2, int i3, int i4,
Bundle bundle) throws IntentSender.SendIntentException {
             super.startIntentSenderForResult(intentSender,   i,
intent, i2, i3, i4, bundle);
    }

    /* access modifiers changed from: protected */
    @Deprecated
    public void onActivityResult(int i, int i2, Intent intent) {
         if (!this.mActivityResultRegistry.dispatchResult(i, i2,
intent)) {
            super.onActivityResult(i, i2, intent);
        }
    }

    @Deprecated
     public  void  onRequestPermissionsResult(int  i,  String[]
strArr, int[] iArr) {
         if (!this.mActivityResultRegistry.dispatchResult(i, -1,
new
Intent().putExtra(ActivityResultContracts.RequestMultiplePermiss
ions.EXTRA_PERMISSIONS,
strArr).putExtra(ActivityResultContracts.RequestMultiplePermissi
ons.EXTRA_PERMISSION_GRANT_RESULTS,           iArr))           &&
Build.VERSION.SDK_INT >= 23) {
            super.onRequestPermissionsResult(i, strArr, iArr);
        }
    }

       public    final    <I,    O>    ActivityResultLauncher<I>
registerForActivityResult(ActivityResultContract<I,           O>
activityResultContract,                  ActivityResultRegistry
activityResultRegistry,                ActivityResultCallback<O>
activityResultCallback) {
         return activityResultRegistry.register("activity_rq#" +
this.mNextLocalRequestCode.getAndIncrement(),              this,
activityResultContract, activityResultCallback);
    }

       public    final    <I,    O>    ActivityResultLauncher<I>
registerForActivityResult(ActivityResultContract<I,           O>
activityResultContract,                ActivityResultCallback<O>
activityResultCallback) {
         return registerForActivityResult(activityResultContract,
this.mActivityResultRegistry, activityResultCallback);
    }
```

```
            public final ActivityResultRegistry
getActivityResultRegistry() {
        return this.mActivityResultRegistry;
    }

        public void onConfigurationChanged(Configuration
configuration) {
        super.onConfigurationChanged(configuration);
                    Iterator<Consumer<Configuration>> it =
this.mOnConfigurationChangedListeners.iterator();
        while (it.hasNext()) {
            it.next().accept(configuration);
        }
    }

                    public final void
addOnConfigurationChangedListener(Consumer<Configuration>
consumer) {
        this.mOnConfigurationChangedListeners.add(consumer);
    }

                    public final void
removeOnConfigurationChangedListener(Consumer<Configuration>
consumer) {
        this.mOnConfigurationChangedListeners.remove(consumer);
    }

    public void onTrimMemory(int i) {
        super.onTrimMemory(i);
                        Iterator<Consumer<Integer>> it =
this.mOnTrimMemoryListeners.iterator();
        while (it.hasNext()) {
            it.next().accept(Integer.valueOf(i));
        }
    }

    public final void addOnTrimMemoryListener(Consumer<Integer>
consumer) {
        this.mOnTrimMemoryListeners.add(consumer);
    }

                    public final void
removeOnTrimMemoryListener(Consumer<Integer> consumer) {
        this.mOnTrimMemoryListeners.remove(consumer);
    }

    /* access modifiers changed from: protected */
    public void onNewIntent(Intent intent) {
        super.onNewIntent(intent);
```

```
        Iterator<Consumer<Intent>>              it              =
this.mOnNewIntentListeners.iterator();
        while (it.hasNext()) {
            it.next().accept(intent);
        }
    }

     public  final  void  addOnNewIntentListener(Consumer<Intent>
consumer) {
        this.mOnNewIntentListeners.add(consumer);
    }

    public final void removeOnNewIntentListener(Consumer<Intent>
consumer) {
        this.mOnNewIntentListeners.remove(consumer);
    }

    public void onMultiWindowModeChanged(boolean z) {
        if (!this.mDispatchingOnMultiWindowModeChanged) {
              Iterator<Consumer<MultiWindowModeChangedInfo>> it =
this.mOnMultiWindowModeChangedListeners.iterator();
            while (it.hasNext()) {
                                                it.next().accept(new
MultiWindowModeChangedInfo(z));
            }
        }
    }

    /* JADX INFO: finally extract failed */
       public    void    onMultiWindowModeChanged(boolean    z,
Configuration configuration) {
        this.mDispatchingOnMultiWindowModeChanged = true;
        try {
            super.onMultiWindowModeChanged(z, configuration);
            this.mDispatchingOnMultiWindowModeChanged = false;
              Iterator<Consumer<MultiWindowModeChangedInfo>> it =
this.mOnMultiWindowModeChangedListeners.iterator();
            while (it.hasNext()) {
                                                it.next().accept(new
MultiWindowModeChangedInfo(z, configuration));
            }
        } catch (Throwable th) {
            this.mDispatchingOnMultiWindowModeChanged = false;
            throw th;
        }
    }

                  public                final                void
addOnMultiWindowModeChangedListener(Consumer<MultiWindowModeChan
gedInfo> consumer) {
```

```
        this.mOnMultiWindowModeChangedListeners.add(consumer);
    }

                    public                final                void
removeOnMultiWindowModeChangedListener(Consumer<MultiWindowModeC
hangedInfo> consumer) {
         this.mOnMultiWindowModeChangedListeners.remove(consumer)
;
    }

    public void onPictureInPictureModeChanged(boolean z) {
        if (!this.mDispatchingOnPictureInPictureModeChanged) {
              Iterator<Consumer<PictureInPictureModeChangedInfo>>
it = this.mOnPictureInPictureModeChangedListeners.iterator();
            while (it.hasNext()) {
                                                it.next().accept(new
PictureInPictureModeChangedInfo(z));
            }
        }
    }

    /* JADX INFO: finally extract failed */
      public   void   onPictureInPictureModeChanged(boolean   z,
Configuration configuration) {
        this.mDispatchingOnPictureInPictureModeChanged = true;
        try {
                              super.onPictureInPictureModeChanged(z,
configuration);
                this.mDispatchingOnPictureInPictureModeChanged  =
false;
              Iterator<Consumer<PictureInPictureModeChangedInfo>>
it = this.mOnPictureInPictureModeChangedListeners.iterator();
            while (it.hasNext()) {
                                                it.next().accept(new
PictureInPictureModeChangedInfo(z, configuration));
            }
        } catch (Throwable th) {
                this.mDispatchingOnPictureInPictureModeChanged  =
false;
            throw th;
        }
    }

                    public                final                void
addOnPictureInPictureModeChangedListener(Consumer<PictureInPictu
reModeChangedInfo> consumer) {
         this.mOnPictureInPictureModeChangedListeners.add(consume
r);
    }
```

```
  public                          final                          void
removeOnPictureInPictureModeChangedListener(Consumer<PictureInPi
ctureModeChangedInfo> consumer) {
         this.mOnPictureInPictureModeChangedListeners.remove(cons
umer);
    }

    public void reportFullyDrawn() {
        try {
            if (Trace.isEnabled()) {
                    Trace.beginSection("reportFullyDrawn()  for
ComponentActivity");
            }
            if (Build.VERSION.SDK_INT > 19) {
                super.reportFullyDrawn();
                  }  else  if  (Build.VERSION.SDK_INT  ==  19  &&
ContextCompat.checkSelfPermission(this,
"android.permission.UPDATE_DEVICE_STATS") == 0) {
                super.reportFullyDrawn();
            }
            this.mFullyDrawnReporter.fullyDrawnReported();
        } finally {
            Trace.endSection();
        }
    }

     private  ReportFullyDrawnExecutor  createFullyDrawnExecutor()
{
        if (Build.VERSION.SDK_INT < 16) {
            return new ReportFullyDrawnExecutorApi1();
        }
        return new ReportFullyDrawnExecutorApi16Impl();
    }

    static class Api19Impl {
        private Api19Impl() {
        }

        static void cancelPendingInputEvents(View view) {
            view.cancelPendingInputEvents();
        }
    }

    static class Api33Impl {
        private Api33Impl() {
        }

                              static       OnBackInvokedDispatcher
getOnBackInvokedDispatcher(Activity activity) {
            return activity.getOnBackInvokedDispatcher();
```

```
        }
    }

      static   class   ReportFullyDrawnExecutorApi1   implements
ReportFullyDrawnExecutor {
        final Handler mHandler = createHandler();

        public void activityDestroyed() {
        }

        public void viewCreated(View view) {
        }

        ReportFullyDrawnExecutorApi1() {
        }

        public void execute(Runnable runnable) {
            this.mHandler.postAtFrontOfQueue(runnable);
        }

        private Handler createHandler() {
            Looper myLooper = Looper.myLooper();
            if (myLooper == null) {
                myLooper = Looper.getMainLooper();
            }
            return new Handler(myLooper);
        }
    }

       class    ReportFullyDrawnExecutorApi16Impl    implements
ReportFullyDrawnExecutor,        ViewTreeObserver.OnDrawListener,
Runnable {
                    final     long     mEndWatchTimeMillis     =
(SystemClock.uptimeMillis() + WorkRequest.MIN_BACKOFF_MILLIS);
        boolean mOnDrawScheduled = false;
        Runnable mRunnable;

        ReportFullyDrawnExecutorApi16Impl() {
        }

        public void viewCreated(View view) {
            if (!this.mOnDrawScheduled) {
                this.mOnDrawScheduled = true;
                 view.getViewTreeObserver().addOnDrawListener(thi
s);
            }
        }

        public void activityDestroyed() {
```

```
ComponentActivity.this.getWindow().getDecorView().removeCallback
s(this);
             ComponentActivity.this.getWindow().getDecorView().ge
tViewTreeObserver().removeOnDrawListener(this);
        }

        public void execute(Runnable runnable) {
            this.mRunnable = runnable;
                                        View      decorView      =
ComponentActivity.this.getWindow().getDecorView();
            if (!this.mOnDrawScheduled) {
                                       decorView.postOnAnimation(new
ComponentActivity$ReportFullyDrawnExecutorApi16Impl$$ExternalSyn
theticLambda0(this));
                            }   else   if   (Looper.myLooper()   ==
Looper.getMainLooper()) {
                decorView.invalidate();
            } else {
                decorView.postInvalidate();
            }
        }

        /* access modifiers changed from: package-private */
          /*  renamed  from:  lambda$execute$0$androidx-activity-
ComponentActivity$ReportFullyDrawnExecutorApi16Impl   reason: not
valid java name */
                     public     /*     synthetic     */     void
m3lambda$execute$0$androidxactivityComponentActivity$ReportFully
DrawnExecutorApi16Impl() {
            Runnable runnable = this.mRunnable;
            if (runnable != null) {
                runnable.run();
                this.mRunnable = null;
            }
        }

        public void onDraw() {
            Runnable runnable = this.mRunnable;
            if (runnable != null) {
                runnable.run();
                this.mRunnable = null;
                                                                  if
(ComponentActivity.this.mFullyDrawnReporter.isFullyDrawnReported
()) {
                    this.mOnDrawScheduled = false;
                     ComponentActivity.this.getWindow().getDecorV
iew().post(this);
                }
                    }  else  if  (SystemClock.uptimeMillis()  >
this.mEndWatchTimeMillis) {
```

```
                this.mOnDrawScheduled = false;
                ComponentActivity.this.getWindow().getDecorView(
).post(this);
            }
        }

        public void run() {
             ComponentActivity.this.getWindow().getDecorView().ge
tViewTreeObserver().removeOnDrawListener(this);
        }
    }
}
```