СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В современном обществе стремительными темпами развиваются информационные и мобильные технологии, и все большее влияние на нашу жизнь оказывает мобильная техника и интернет.

Теперь безграмотным человеком считается не тот, кто не умеет читать или писать, а тот, кто не умеет пользоваться мобильным телефоном, компьютером или интернетом, но таких людей с каждым годом становиться все меньше [30].

С появлением интернета на него сразу же обратил внимание бизнес, так как это место могло стать хорошей нищей для сбыта продукции и услуг. Обороты компаний увеличились в десятки и сотни раз, и это повлияло на дальнейшее развитие технологий и революционному созданию мобильных устройств.

С помощью мобильных технологий имеется возможность заплатить за парковку прямо с телефона. Встроенные карты позволяют не заблудиться в любом районе незнакомого города.

Сейчас доступны расписания транспорта по нажатию одной лишь кнопки.

Фактически для сотен миллионов людей стало совершенно естественным и привычным сразу искать в мобильных устройствах и планшетах информацию о любой деятельности.

Гаджеты уже могут предвидеть, какая информация может заинтересовать, и подают её в удобной форме. Подобное, на первый взгляд, поверхностное и легкомысленное повседневное использование мобильных технологий изменило человеческое сообщество.

Но новые возможности принесли с собой и новые трудности. У каждой технологии есть свои преимущества и недостатки, с которыми приходится либо мириться, либо искать пути их решения.

Потенциал разрабатываемых мобильных приложений готов заменить бумаги для учета заказов, и освободило бы человека от рутинной работы формирования статистики заказов.

Разработка такого рода приложений учитывает специфику заказов.

Объектом исследования дипломной работы является мобильное-приложение для учета заказов и формирования статистики «Time management».

Предмет исследования: особенности учета заказов и формирования статистики на мобильных устройствах.

Целью дипломной работы является проектирование и разработка мобильного приложения «Time management».

Для достижения цели необходимо решить следующие задачи:

– рассмотрение аналогов разрабатываемого мобильного приложения, их ключевых достоинств, которые необходимо перенять, и недостатков, которые следует устранить;

– теоретический анализ и обработка практической и методической литературы по дипломной работе;

– рассмотрение классификации информационных систем и принципа автоматизации бизнес-процессов;

– выявление требований к информационной системе и постановка задачи на разработку информационной системы;

– проектирование и реализация приемлемой для учебного отдела автоматизированной системы составления расписания.

– анализ инструментов необходимых для реализации задачи, их достоинства и недостатки.

Практическая значимость данной работы заключается в том, что полученное, в результате исследования, мобильное приложение будет иметь возможность применения в деятельности.организаций занимающиеся учетом заказов и формирования статистки [4,5].

1. ОБЩАЯ ЧАСТЬ

1.1 Описание предметной области

Штамп - ручной инструмент, изготовленный из различных видов материалов, способный выполнять функцию фиксирования события в виде оттиска или переноса красителя на различные материалы: глина, металл, воск, сургуч, кожа, бумага, а также тесто [4].

Штамповка - процесс пластической деформации материала с изменением формы и размеров тела. основные вида штамповки - листовая и объёмная;

– листовая штамповка - подразумевает в исходном виде тело, одно из измерений которого пренебрежимо мало по сравнению с двумя другими Примером листовой штамповки является процесс пробивания листового металла, в результате которого получают перфорированный металл;

– объемная штамповка - процесс, заключающийся в изменении простейших объёмных заготовок цилиндрической, призматической и других форм, в более сложные изделия, форма которых соответствует полости специализированных инструментов [16].

Также существуют штампы стилизованные под предметы - штампы-авторучки, штамп-брелок и карманные штампы:

– штампы-авторучки - ручка в который находится миниатюрный механизм под клише штампа, позволяет легко сделать позволяет легко передать контактную информацию о себе;

– штамп-брелок - брелок со встроенным в корпус красконаполненным штампом. Штамп-брелоки могут иметь различные формы и размеры: от классических круглых и овальных до более сложных и оригинальных. На них можно наносить различные надписи, логотипы, изображения.

– карманный-штамп - представляет из себя карманный штамп к которому прикреплено клише квадратной или круглой формы. Такие штампы чаще всего используют врачи, нотариусы, адвокаты [17].

Штампы бывают разных форм:

– прямоугольные обычно с указанием реквизитов учреждения -наименование, адрес и тому подобное. Чаще всего используется предприятиями на производстве, а также для быстрого заполнения своих реквизитов в различной документации. Физические лица используют штамп в основном для товарных чеков или для проставления текста рекламного характера;

– круглые используются для маркировки документов, для пропускных документов, для пропусков и удостоверений;

– треугольные как правило, используемые для справок и внутренних документов или иных, порою вычурных форм в виде фигур;

– квадратные применяются для маркировки различных типов товаров, а также для рекламных и маркетинговых целей [17].

Оснастка - это корпус штампа, за который надо держаться и на который крепится клише. Оснастки разделяются на ручные автоматические и флэш.

– автоматическая оснастка - это изделие, в основе работы которого заложен определенный механизм, обеспечивающий перенос краски со встроенной подушки посредством печатающего элемента - клише. Автоматические оснастки имеют уже встроенную штемпельную подушку и позволяют сделать большое количество оттисков за короткое время без особых хлопот. Еще одним важным достоинством является получение равномерного и четкого оттиска, отсутствие отдельной штемпельной подушки по сравнению с ручными. Оснастки автоматические бывают сделаны из металла или из пластика. Пластиковые легкие и наиболее популярны из-за цены. Металлические дороже, но их конструкция рассчитана на интенсивное использование и на большое количество высококачественных оттисков [24];

– ручная оснастка - это рукоятка с площадкой определенного размера и формы, к которой крепится печатающий элемент - клише. Для их использования нужна штемпельная подушка с краской. Есть модели в, которых сразу имеется штемпельная подушка и их удобно брать с собой. Ручные оснастки сегодня в основном разделяют по материалу из которого они сделаны: это либо пластик,

либо металл. Это не плохой выбор, если не требуется ежедневно делать большое количество оттисков. Сверху головки расположен прозрачный колпачок, под которым находится образец оттиска штампа [17];

– флэш - составные элементы такой оснастки: корпус, картридж, микропористая резина, стопорная рамка или кольцо. У них нет вращательного механизма как у автоматических, или подушки как у ручных, так как краска сразу помещается в печатающий элемент микропористую резину и картридж. По сроку службы флэш оснастки не уступают автоматическим. Эта оснастка бывает изготовлена из пластика и называется пластиковой флэш оснасткой, изготовленная из металла называется металлическая флэш оснастка [16].

Клише - рельефная поверхность макета печати. Изготавливается из различных материалов композит, полимер, резина и закрепляется на оснастке для комфортного проставления оттиска - чернильного следа на бумаге.

Виды клише:

– композитные клише - композитные материалы позволяют создать клише с элементами повышенной сложности и защитой от дублирования. Тончайшие элементы, сетки и даже фотографии можно выполнить только с помощью композита;

– полимерные - клише из полимера изготавливается максимально быстро. Уже через 2 часа вы сможете проставить оттиск, при условии, что макет не содержит сложных и экстратонких элементов. Отлично подойдет для изготовления стандартных штампов;

– резиновые - создается с помощью лазерной гравировки и имеет хорошие эксплуатационные качества. Подходит для использования спиртовых красок, масляных и на водной основе;

– гербовые - клише изготавливается с соблюдением требований ГОСТ Р 51551-2001. Обязательный для изготовления материал - резина. Такое клише проходит обязательную регистрацию и проверку в реестре изготовителя [24].

Существуют несколько основных цветов чернил для штампов:

– чёрный;

– красный;

– синий;

– зелёный;

– фиолетовый [24].

Если провести анализ нормативных правовых актов РФ, в которых идет речь об организации документооборота, прямого запрета на использование каких-либо цветов оттиска печати нет. Соответственно, организации могут выбирать цвет оттиска по своему усмотрению. Однако нужно иметь в виду, что нормативные правовые акты устанавливают необходимость проставления четкой, хорошо различимой печати, чтобы не возникало сомнения в подлинности заверяемого ею документа. Если, например, на бланке весь текст набран чёрным цветом, то ставить такую же печать - неправильно, так как она будет трудноразличимой, это нарушение стандартов. В этом случае лучше предпочесть синюю или фиолетовую [8].

В российском законодательстве предельно четко определены изображения, которые могут быть использованы в печати, но не цвета.

Например, коммерческие организации, не могут использовать гербовые печати и штампы их применяют только государственные организации, в числе которых не только органы власти.

Но и в этом случае ничего о цвете оттиска печати в нормативных правовых актах не указывается [17].

Виды печатей по материалу изготовления:

– металлические печати - предназначены для опечатывания дверей, сейфов, металлических шкафов, банковских хранилищ, а также прочих ёмкостей, шкафов и т. д., путём проставления оттиска на пластилине, сургуче или других более мягких по сравнению с пломбиратором материалах. Применяются совместно с опечатывающими устройствами. Изготавливаются на заготовках с

кольцом для ключей диаметром от 20 мм, оттиск наносится путём механической гравировки печати;

– резиновые - дорогостоящий, но один из самых филигранных видов печати. С помощью лазерной гравировки можно создавать мельчайшие детали и сложные рисунки для оттиска. Резиновые печати применяются как в делопроизводстве, так и в творческих областях для нанесения повторяющихся узоров и принтов. Вулканизация удешевляет изготовление резиновых печатей. В качестве матрицы применяют форму из фотополимера;

– полимерные - хорошо переносят высокие нагрузки и давление, благодаря чему они отличаются высокой износостойкостью. С осторожностью применяйте чернила на спиртовой основе для этого типа печатей. Это доступный вариант устройств, поскольку имеют недолгий цикл производства и низкую стоимость обслуживания [17];

– красконаполненные или флеш-печати - устройство этой печати позволяет обходиться без штемпельной подушки. Пористую резину пропитывают масляной краской, как губку, оставляют бесшумный и чёткий оттиск, в котором детально прорисованы мелкие элементы и тонкие линии. Оттиск не растекается, быстро сохнет, его не смоешь водой. Секрет заключается в том, что такие печати делают по новой технологии: пористую резину пропитывают масляной краской, как губку. Обратите внимание, что заправлять эти печати красками на водной или спиртовой основе не рекомендуется [24].

Полиграфия - отрасль промышленности, занимающаяся изготовлением книжно-журнальной, деловой, газетной, этикеточной, картографической, упаковочной, акцидентной и прочей печатной продукции [4].

Виды печатей в полиграфии:

– офсетная - один из лидирующих способов печати. Позволяет изготавливать листовую и рулонную продукцию крупными тиражами;

– прямая - подходит для небольших тиражей и печати на тканях, металле, дереве, стекле, металле, акриле;

– литография - на камень, который могут заменять бумагой, цинком или алюминием, предварительно наносится изображение карандашом. Путем оттиска его затем переносят на носитель;

– струйная - это метод печати, который использует струи краски, чтобы создавать изображения на различных материалах. Этот метод печати широко используется в производстве фотографий, баннеров, этикеток и других видов рекламы. предназначены для сверхмалых тиражей. Такая печать используется для производства дисков CD и DVD, газет, книг [8];

– цифровая - это метод печати, который использует цифровую технологию для создания изображений на различных материалах. Она широко используется в рекламе, упаковке, изготовлении наружной рекламы, этикеток и т.д. Подходит для создания любого объема буклетов, визиток, бланков и пр. Применяется также для нанесения изображения на ткани [16];

– широкоформатная - это метод печати, который используется для создания больших изображений на различных материалах. Она широко применяется в рекламе, дизайне интерьера, производстве наружной рекламы, баннеров, постеров и т.д. Повсеместно используется в наружной рекламе и изготовлении элементов интерьера, которые создаются малыми тиражами;

– сублимационная - это метод печати, который использует специальные чернила, которые при нагревании превращаются в газ и проникают в поверхность материала. Часто применяется для изготовления сувенирной продукции самого разного назначения, печати на тканях, только синтетических;

– ультрафиолетовая - применяется для нанесения на самые разные материалы небольшими тиражами;

– шелкография - это метод печати, который использует шелковые экраны для передачи изображения на поверхность материала. Этот метод печати широко используется в производстве различных предметов, таких как футболки, кружки, баннеры и т.д. [17];

– флексография - применятся при изготовлении этикеток, бирок, конвертов, одноразовой посуды, листовок и др. Более выгодна при тиражах [16];

Типы печатей и штампов по назначению:

– для компаний - содержит информацию о компании, ее названии, адресе и контактных данных. Он используется для создания копий всех документов, связанных с работой компаний, служит для идентификации, является главной печатью предприятия.

– печать индивидуального предпринимателя - используется бизнесменом для заключения сделок, подтверждения подлинности и защиты внешних и внутренних документов. Оттиск содержит номер ОГРНИП, ФИО предпринимателя, а также логотип и слоган;

– государственная печать - используется в муниципальных и госучреждениях. Отличаются тем, что оттиск содержит элементы государственной символики. Также содержание клише строго регламентировано. Печать медицинской клиники. Обычно её держат у главного врача. Информация на печати касается её названия, регистрационного номера и места регистрации. В медицинских организациях используется преимущественно треугольная печать [17];

– для медицинских организаций - используется в больницах, бывают двух видов [16];

– врачебная печать - являются важным элементом в медицинской практике, используемым для подтверждения подписи врача и его квалификации. Она используется для подписи всех медицинских документов, включая направления на обследование и лечение, рецепты и справки.

– печать мед. организации - штамп медицинской организации, имеет треугольную форму и также размер в 40 мм. В центре такой печати чаще всего написано, что эта печать используется для подтверждения подлинности различных документов. По краям треугольника наносят название медицинского государственного учреждения, дальше на штампе добавляют регистрационный номер и в каком месте этот регистрационный номер был поставлен.

– нотариальная печать - используется нотариальными службами для заверения документов. Нотариальной печатью подтверждается подлинность документов расписок, завещаний, распоряжений, доверенностей, договоров купли/продажи и их юридическая сила;

– факсимиле - используется для подписания документов, факсимиле -это точное воспроизведение графического оригинала: подпись, рисунок, чертеж. Такие штампы нашли большое применение в личных домашних библиотеках. Оснастка для факсимильного штампа, как правило, традиционная - прямоугольной или треугольной формы, а оттиск может быть каким угодно: подпись, вензель, родовой герб - в зависимости от решения владельца штамп. Они могут быть полезными в медицинской практике, особенно при необходимости создания копий важных медицинских документов [24].

1.2 Обзор аналогов

В настоящее время существует огромное количество мобильных приложений для учета заказов и формирования статистики. Данные приложения также можно использовать для оформления заказов связанных с штампами либо печатями, так как они имеют оплату, предоплату, дедлайны, информацию о клиентах и другие атрибуты заказов. Все приложения справляются со своей задачей - составление статистики по внесенным заказам. Поэтому для обзора аналогов были взяты обычные приложения для учета заказов и формирования статистки [11].

«ProBiznes» - мобильное приложения для учета заказов и формирования статистики по внесенным заказам, используется маленькими компаниями, обычными людьми так как находится в свободном доступе, но имеют платный контент. Имеет низкие оценки на порталах скачивания мобильных приложений. Данное мобильное приложение позволяет записывать заказы, их дедлайны, записывать клиентов и имеет общий доступ к заказам на разных мобильных устройствах. Интерфейс мобильного приложения предоставлен на рисунке 1 [3].

Рисунок 1 -Интерфейс мобильного приложения «ProBiznes»

Преимущества:

– функция добавления постоянных клиентов;

– назначение сотрудника для выполнения заказа;

– составление полной статистики с учетом предоплаты.

Недостатки:

– не может учесть специфику заказа;

– нет функции комментирования заказа;

– неоптимизированное приложения под различные версии операционных систем мобильных приложений, долгая загрузка [26].

«BiznesMaxis» - это приложение которое так же находится в свободном доступе полностью. бесплатное и в полне подходит для учета заказов и формирования статистики с удобным, используется маленькими компаниями, обычными людьми так как находится в свободном доступе, но имеют платный контент. Данное мобильное приложение позволяет записывать заказы просматривать задачи в календаре.

Интерфейс мобильного приложения предоставлен на рисунке 2.

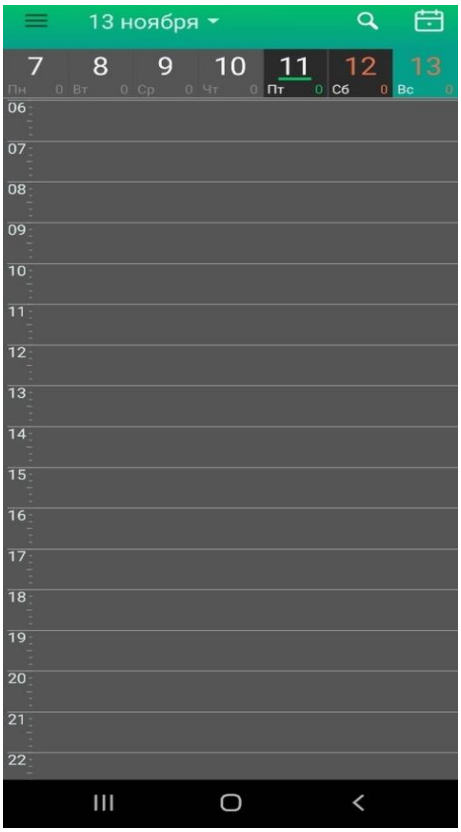

Рисунок 2 -Главный экран мобильного приложения «BiznesMaxis»

Преимущества:

– удобная заполняемая форма заказа;

– функция напоминания о заказе за устанавливаемое время;

– функция метка для передачи заказа клиенту;

– настраиваемый дизайн и заголовки для каждого заказа.

Недостатки:

– нет интерфейса отображения всех заказов;

– дизайн приложение не детализирован

– неоптимизированное приложения под различные версии операционных систем мобильных приложений [26].

«Все сам» - простое мобильное приложения для учета заказов с возможностью добавления партнеров, созданию отчетов, созданию прайс листа для быстрого доступа к предлагаемым услугам, все функции в приложении настраиваются и имеют приятный дизайн. Возможность вести базу клиентов, создания наглядных диаграмм по статусам.

На рисунке 3 показан интерфейс данного приложения [11].

Рисунок 3 -Интерфейс мобильного приложения «Все сам»

Преимущества:

– удобная форма составления заказа;
– возможность добавить партнера для выполнения общих заказов;
– функция добавления плана выполнения заказа;
– создание наглядных диаграмм по статусам заказов;
– добавление своего прайса для быстрой озвучки для клиента.

Недостатки:

– неудобный дизайн приложения;
– выплывающие ненужные строки с информацией.
– неадаптированное приложение для разных мобильных устройств [26].

2. СПЕЦИАЛЬНАЯ ЧАСТЬ

2.1 Аналитическая часть

2.1.1Постановка задачи

Мобильное приложение для учета заказов и формирования статистки «Time management» основано на базе операционной системы Android.

Мобильное приложение для учета заказов и формирования статистки «Time management» приносит удобство при ведении учета заказов и автоматически формирует статистку на основе введенных в приложение заказов во вкладке статистка, имеется удобный фильтр по датам с возможностями точного выбора даты. При использовании приложения пользователь имеет быстрый доступ к информации о заказах и статистке, и имеет возможность добавлять новые заказы [10].

Мобильное приложение может помочь уменьшить ошибки в учете заказов и формировании статистики благодаря возможности проверки на ошибки. Данное мобильное приложение будет иметь следующие функции: добавление новых заказов, при нажатие кнопки «+» во вкладке «Заказы»;

При регистрирование заказа в приложении пользователю необходимо заполнить следующие поля:

- заказчик;
- город;
- количество;
- общая сумма заказа;
- оплачено, аванс на заказ;
- дата принятия заказа;
- заметки, комментарий на заказ;
- формирование статистки;
- фильтрация статистики по датам;
- редактирование созданных заказов [1,3].

2.1.2Разработка алгоритма мобильного приложения

Разработка мобильного приложения «Time management» включает следующие этапы:

- создание дизайна мобильного приложения;
- разработка активностей и связь между ними;
- разработка базы данных мобильного приложения;
- интеграция с базой данных;
- тестирование мобильного приложения;
- отладка мобильного приложения [1,27,28].

2.1.2.1 Разработка блок схемы

Использование мобильного приложение осуществляется любым человеком с данной программой на мобильном устройстве. Для разработки алгоритмов мобильного приложения будут использоваться блок-схемы [22].

Блок-схема - это схематичное представление процесса, системы или компьютерного алгоритма. Блок-схемы часто применяются в разных сферах деятельности, чтобы документировать, изучать, планировать, совершенствовать и объяснять сложные процессы с помощью простых логичных диаграмм.

Для построения блок-схем применяются прямоугольники, овалы, ромбы и некоторые другие фигуры а также соединительные стрелки, которые указывают последовательность шагов или направление процесса [28].

Блок-схемы варьируются от простых, нарисованных вручную до подробных, составленных на компьютере диаграмм со множеством шагов и процессов. Если учесть все возможные вариации, блок-схемы можно признать одним из самых распространенных видов схем во всем мире. Они широко используются в разных сферах как технической, так и нетехнической направленности моделирования [7].

Иногда блок-схемы получают более узкоспециальные названия, например, схема процесса, схема рабочего процесса, функциональная блок-схема,

моделирование бизнес-процессов, модель и нотация бизнес-процессов BPMN или схема технологического процесса PFD.

Они тесно связаны с другими распространенными видами схем, такими как диаграммы DFD и диаграммы активности на унифицированном языке моделирования UML.

На рисунке 4 представлена блок схема алгоритма использования мобильного приложения «Time management» [5,6].

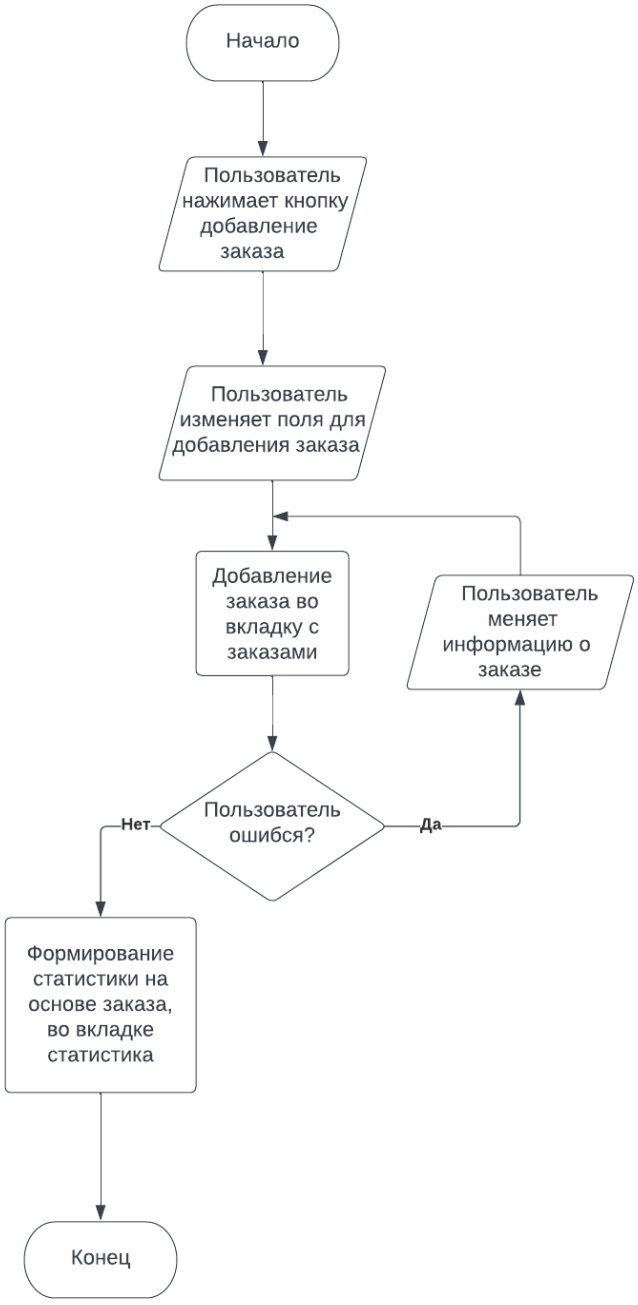

Рисунок 4 -Блок схема алгоритма пользования приложением «Time management»

Пользователь:

– вводит информацию о заказе, есть много полей для ввода - заказчик, город, количество заказов, общая сумма, оплачено, дата, заметки;

– просматривает и при необходимости меняет информацию о заказе во вкладке заказы;

– просматривает общую статистку о заказах во вкладке статистика, используя фильтры.

Приложение:

– делает возможным добавление заказов;

– осуществляет режим редактирования заказов, при необходимости;

– формирует общую статистику о заказе;

– предоставляет удобную фильтрацию по датам либо по месяцам.

2.1.2.2 Разработка диаграмм вариантов использования

Исходя из составленной блок-схемы алгоритма использования мобильного приложения «Time management» можно выделить его возможности использования различными типами пользователей. Для этого будет применена Use Case диаграмма или же диаграмма вариантов использования.

Use case диаграмма - это графическое представление функциональных требований к системе. Она отображает взаимодействие между пользователями и системой, а также функциональные возможности, которые система предоставляет для каждого актера.

Use case диаграмма помогает определить функциональные требования к системе и установить связи между пользователями и функциями. Она может быть использована как основа для разработки более подробных диаграмм, таких как диаграммы последовательности и диаграммы классов.

На рисунке 5 представлена диаграмма вариантов использования мобильного приложения «Time management» [9].

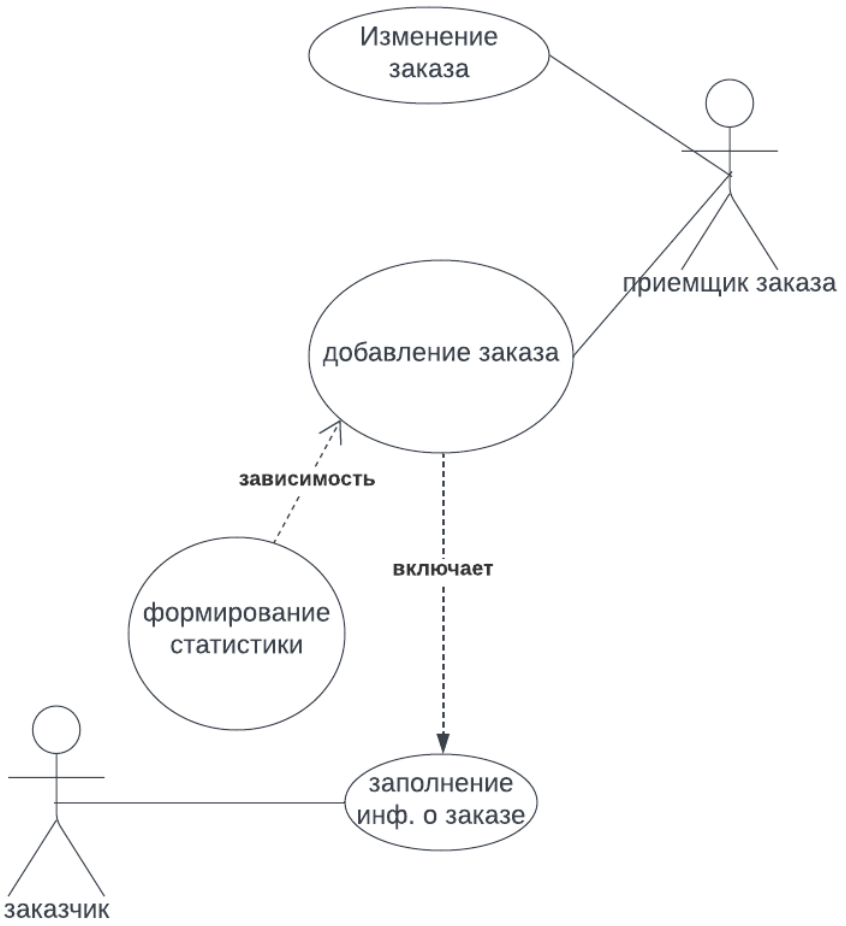

Рисунок 5 - Диаграмма вариантов использования приложения

На данной диаграмме представлены зависимости и включения функций мобильного приложения по отношению к самим функциям к актерам [28].

2.1.2.3 Разработка диаграммы базы данных

База данных, являющаяся одной из важнейших частей информационной системы, конечно же, представляет собой сложный объект, который также подлежит проектированию.

В процессе создания информационной системы проектирование базы данных имеет очень важную роль, так как база данных является фундаментом информационной системы. Проектирование базы данных выполняется после анализа требований к будущей системе. А уже после того, как выработана общая схема базы данных, происходит процесс определения архитектуры будущей информационной системы [21,31].

Таким образом, проектирование базы данных играет огромную роль в создании будущей информационной системы, являясь её фундаментом [31].

На рисунке 6 представлена схема спроектированной базы данных.

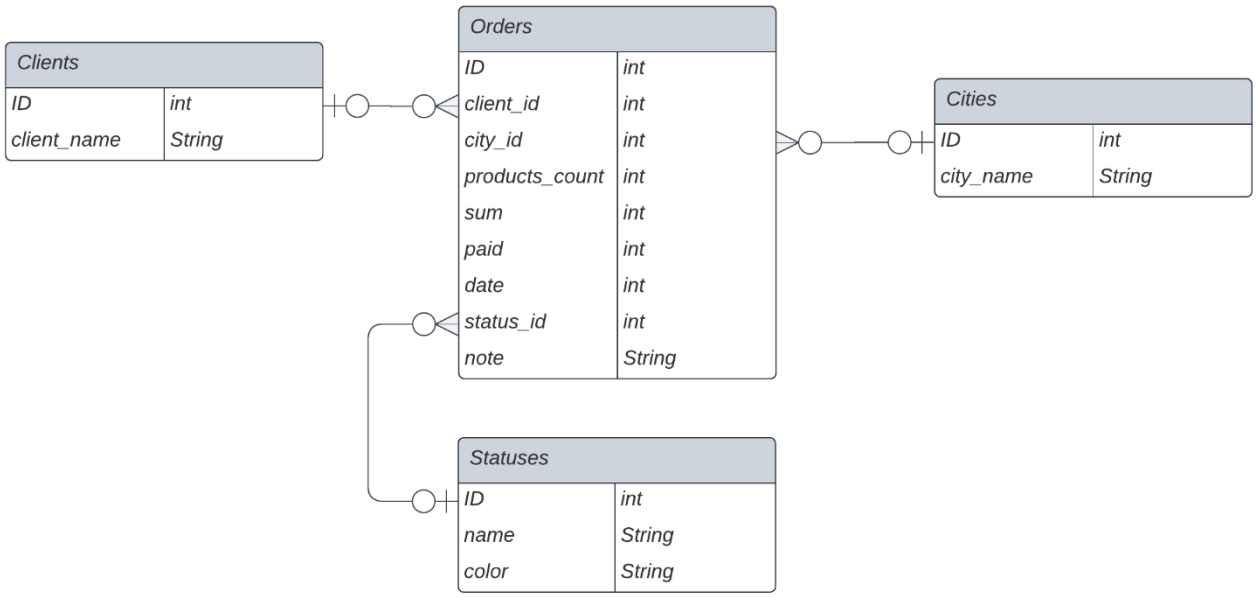

Рисунок 6 -Схема спроектированной базы данных

2.1.3Обоснование выбора языка программирования

Для разработки мобильного приложения для учета заказов и формирования статистики будет использоваться язык программирования Java.

Java - строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems. Разработка ведётся сообществом, организованным через Java Community Process; язык и основные реализующие его технологии распространяются по лицензии GPL. Права на торговую марку принадлежат корпорации Oracle [15,20,27].

Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, для которой существует реализация виртуальной Java-машины. Дата официального выпуска - 23 мая 1995 года [25].

Ключевой особенностью языка Java является то, что его код сначала транслируется в специальный байт-код, независимый от платформы. А затем байт-код выполняется виртуальной машиной JVM [13].

Подобная архитектура обеспечивает кроссплатформенность и аппаратную переносимость программ на Java, благодаря чему подобные программы без перекомпиляции могут выполняться на различных платформах - Windows, Linux, Mac OS . Для каждой из платформ может быть своя реализация виртуальной машины JVM, но каждая из них может выполнять один и тот же код.

Java является языком с Си-подобным синтаксисом. Для разработки на языке программирования Java потребуется специальный комплект инструментов, который называется JDK или Java Development Kit [13,23].

Еще одной ключевой особенностью Java является то, что данный язык программирования поддерживает автоматическую сборку мусора.

Java является объектно-ориентированным языком. Он поддерживает полиморфизм, наследование, статическую типизацию. Объектно-ориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений.

Достоинства:

– независимость - ваш код будет работать на любой платформе, которая поддерживает Java;

– надёжность - в немалой мере достигается благодаря строгой статической типизации;

– мультифункциональность;

– сравнительно простой синтаксис;

– Java - основной язык для Android-разработки;

– объектно-ориентированное программирование;

– автоматическая сборка мусора [15].

Недостатки:

– низкая скорость;

– требует много памяти;

– нет поддержки низкоуровневого программирования;

– в коммерческих целях, платный.

2.1.4Обоснование выбора инструментальных средств

Для разработки мобильного приложения «Time management» будут использованы:

– Amazon coretto 11;

– Android SDK API 29;

– СУБД SQLite;

– Room для работы с СУБД [19].

Amazon Correto 11 - это бесплатная многоплатформенная версия пакета средств разработки OpenJDK, готовая к использованию в рабочей среде. Corretto поставляется с долгосрочной поддержкой, которая включает в себя повышение производительности и исправления безопасности.. С помощью Corretto разрабатывается и запускается Java-приложения на популярных операционных системах, включая Linux, Windows и macOS.

Corretto поставляется с поддержкой нескольких платформ, что позволяет запускать его в облаке, на своих серверах или локальной машине. Corretto 11, соответствующий JDK 11.

Java Development Kit состоит частей:

– инструменты для разработки программ;

– средства для запуска программ;

Инструменты для разработки - это различные утилиты, средства безопасности, документация, примеры и так далее;

Средства для запуска программы:

– компилятор Javac - переводит исходный текст в байт-код - набор инструкций, понятный виртуальной Java-машине

– отладчик - программу в режиме отладчика можно остановить и запустить снова, можно узнать значения переменных и вычислить выражения на определённом этапе;

– API Java - это программные компоненты, позволяющие интегрировать различные приложения и веб-сайты [30];

Java Runtime Environment - среда выполнения Java. Программы на Java можно запускать на выполнение, только если установлена JRE. Она действует как посредник между программой и операционной системой, позволяет выполнять программу на разных устройствах и ОС.

Достоинства:

– бесплатное пользование;

– многоплатформенность;

– кросплатформенность;

– эргодинамика.

Недостатки:

– сложность в освоении;

– зависимость нагрузок и остраслевых тенденций;

– требуется мощный процессор.

Среда разработки или IDE - интегрированная среда разработки - специальный программный комплекс, предназначенный для полного цикла написания и тестирования программ на определенном языке.

IntelliJ IDEA - интегрированная среда разработки программного обеспечения для многих языков программирования, для языков JVM, предназначенная для максимальной производительности разработчиков. Выполняет рутинные и повторяющиеся задачи, обеспечивая интеллектуальное завершение кода и его статический анализ. Обеспечивает единообразную работу в Windows, macOS и Linux [19].

Поддерживает такие функции как:

– поддержка синтаксиса - подсветка ключевых слов, авто подстановка, подсветка ошибок;

– навигация - переход к объявлению, поиск использований, поиск по текстовой строке, файлу или названию;

– анализ - иерархия классов, а также свойства и действия класса;

– рефакторинг - переименование классов, свойств и действий;

– визуализация форм - отображению разработчику текущего дизайна определенной формы;

– метапрограммирование - возможность на лету генерировать код на основе метакодов;

– отладчик - возможность ставить breakpoint’ы, отлаживать императивную логику, смотреть watches;

– Language Injection - навигация, рефакторинг, автоподстановка и подсветка синтаксиса lsFusion при использовании в других языках - Java и JasperReports XML.

Достоинства:

– доступность;

– кросплатформенность;

– многозадачность;

– компиляция в байт-код;

– максимальная производительность благодаря JVM;

– метапрограммирование;

– форматирование для Java, Groovy, Scala, HTML, CSS, JavaScript, CoffeeScript, ActionScript, LESS, XML;

– интеграция с серверами приложений, включая Tomcat, TomEE, GlassFish, JBoss, WebLogic, WebSphere, Geronimo, Resin, Jetty и Virgo;

– инструменты для работы с базами данных и SQL файлами;

– интеграция с коммерческими системами управления версиями Perforce, Team Foundation Server, ClearCase, Visual SourceSafe [26].

Недостатки:

– сложность в освоении;

– имеются платные функции;

– высокие системные требования.

СУБД SQLite - это быстрая и легкая встраиваемая однофайловая СУБД, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве. Для работы не требуются сторонние библиотеки или службы [19].

Понятие «встраиваемый» означает, что СУБД не использует парадигму клиент-сервер. Движок SQLite - не отдельно работающий процесс, с которым взаимодействует программа, а библиотека. Позволяет записывать новую и запрашивать существующую информацию, изменять ее, настраивать доступ.

SQLite не имеет сервера. Это значит, что все данные программное обеспечение хранит на одном устройстве. СУБД встраивается в приложение и работает как его составная часть. Если установить на компьютер программу, использующую SQLite, то база данных тоже будет храниться на нем же. Формат базы - один текстовый файл, который можно прочитать на любой платформе. Такой подход повышает производительность и скорость работы [19].

Табличные записи хранятся в едином файле, database file, который находится на том же устройстве, что и программа. Чтобы при работе не возникало ошибок, файл блокируется для сторонних процессов перед записью.

Не требует администрирования и работает на мобильных устройствах, игровых приставках, телевизорах, беспилотных летательных аппаратах, камерах, автомобильных мультимедийных системах.

Перед использованием СУБД не нуждается в сложной настройке или длительной установке. Для решения большинства задач можно пользоваться «из коробки», без установки дополнительных компонентов.

Работать с SQLite можно как с библиотекой или через SQLite3.

SQLite3 - это консольная утилита для работы с SQLite от разработчиков СУБД. Она запускается и работает в командной строке, в терминале операционной системы [19].

По функциональности SQLite3 - программа-клиент для клиент-серверных приложений. С ее помощью можно вводить и передавать запросы к базе данных: создавать, модифицировать, получать или удалять таблицу. Разница в том, что

она обращается не к отдельному процессу-серверу, а ко встроенному в приложение движку SQLite.

Достоинства:

– высокая скорость;
– минимализм;
– надежность;
– нулевая конфигурация;
– занимает малое количество физической памяти;
– доступность;
– кросплатформенность;
– автономность;
– хранение данных в одном файле.

Недостатки:

– ограниченная поддержка типов данных;
– отсутствие хранимых процедур;
– ограничения в применении;
– отсутствие бесплатной техподдержки;
– недостатки с операциями записи;
– отсутствие пользовательского управления, нет возможности управлять связями в таблицах в соответствии с привилегиями.

Room - это библиотека, которая является частью архитектурных компонентов Android. Она облегчает работу с объектами SQLite в приложении, уменьшая объём стандартного кода. Все что необходимо разработчику - это "объяснить" библиотеке как выглядят данные, их структуру и способы взаимодействия с помощью специальных аннотаций. Одновременно точка доступа к БД, таблицы хранения сущностей для нее и набор методов для работы с ней. С Room значительно проще использовать запросы, обновления, передачи и удаления данных из SQLite. Одно из главных преимуществ Room, возможность применять ее, как библиотеку сохраняемости. Например, с ее помощью можно кешировать

данные, чтобы отображать их, даже если устройство не подключено к интернету. Эта возможность будет полезна и для решения других задач. Room в целом упрощает организацию данных и взаимодействие с ними [19,25].

Данная библиотека проверяет SQL-запросы во время компиляции, оптимизирует пути миграции базы данных, помогает улучшать повторяющийся и подверженный ошибкам шаблонный код. Благодаря этому можно будет получать сообщения об изменениях в каждом из мест, где они произошли. В общем код вашего проекта станет более структурированным, а процессы –контролируемыми [25].

Это одна из базовых библиотек, без которой не рекомендуют начинать работу над android-проектом в целом. Она входит в Jetpack - список популярных и часто используемых библиотек. Задача «стартового набора» помочь писать меньше кода, сделать его согласованным с актуальными версиями ОС и методологически правильным. В общем в нем есть все, что критически важно для упрощения работы и Room, в том числе.

Room - поддерживает только 4 типа данных, такие же как и в SQLite. Что ограничивает его спектр применения.

Room состоит из 3 основных компонентов:

– entity - объект таблицы БД;

– dao - предоставляет методы, которое приложение может производить над базой данных;

– database - содержит базу данных и служит основной точкой доступа для базового подключения к постоянным данным приложения.

Достоинства:

– позволяет не писать SQL запаросы;

– простое использование;

– использует LifeData, а соответственно при изменение данных в БД, Room автоматически обновляет данные в графическом интерфейсе приложения.

Недостатки:

– не позволяет оборачивать в LifeData сложные SQL запросы;
– поддерживает только 4 типа данных;
– находится в Beta версии;
– проверка SQL запросов во время компиляции;
– возможность применять как библиотеку для хранения.

Android SDK API 29 - это дополнительный набор инструментов, которые помогают написать код, запустить тестирование и отладку, проверить работу мобильного приложения на различных версиях Android и оценить результат в реальном времени. Также пакет позволяет пользователям получать информацию о состоянии операционной системы, читать логи и выявлять ошибки. Через SDK для Андроид можно восстанавливать программную оболочку и устанавливать сторонние прошивки [5].

Достоинства:

– позволяет выполнять разработку мобильных приложений;
– бесплатный;
– запуск тестирования и отладка;
– возможность устанавливать прошивку;

Недостатки:

– долгое освоение;
– необходимость большого количества вычислительной мощности ПК;
– необходимость большого количества RAM памяти.

2.2 Практическая часть

2.2.1Описание процесса разработки мобильного приложения

Для загрузки интегрированной среды разработки InteliJ IDEA необходимо перейти на официальный сайт разработчиков, перейти в раздел загрузки и начать скачивание. Важно, что разработчики распространяют свою среду разработки на платной основе. Имеется 30-дневный пробный период, чего вполне достаточно для разработки.

На рисунке 7 представлена страница для загрузки InteliJ IDEA.

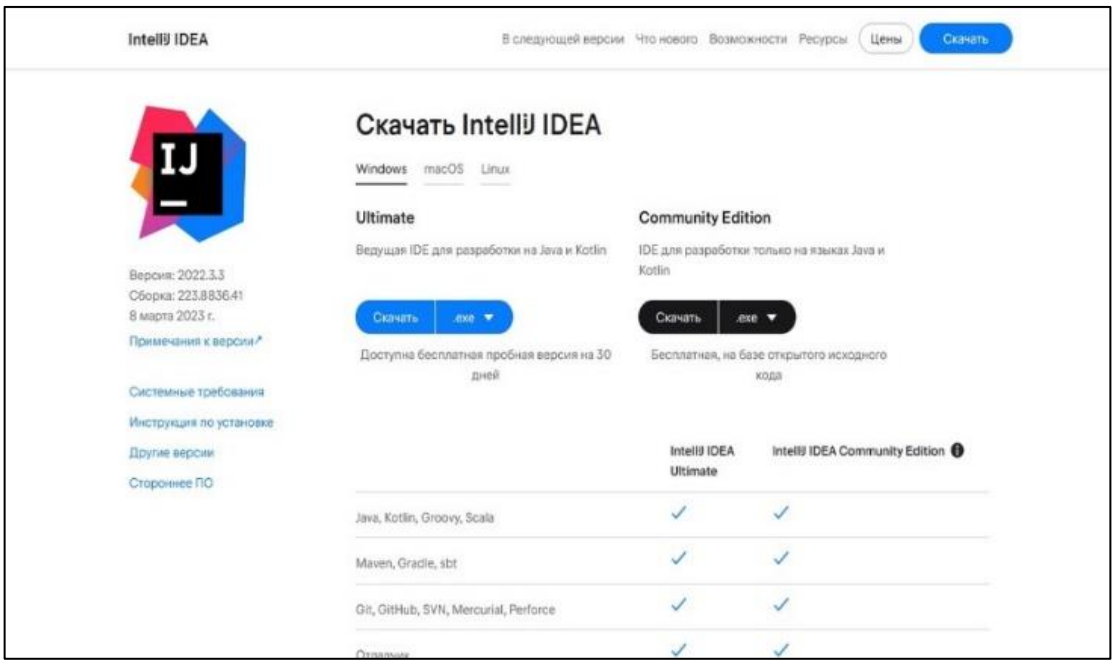

Рисунок 7 - Страница загрузки InteliJ IDEA

После скачивания установочной программы с расширением «exe» необходимо произвести запуск установки интегрированной среды разработки InteliJ IDEA. На рисунках 8 - 9 представлен процесс настройки компиляции среды разработки и установки.

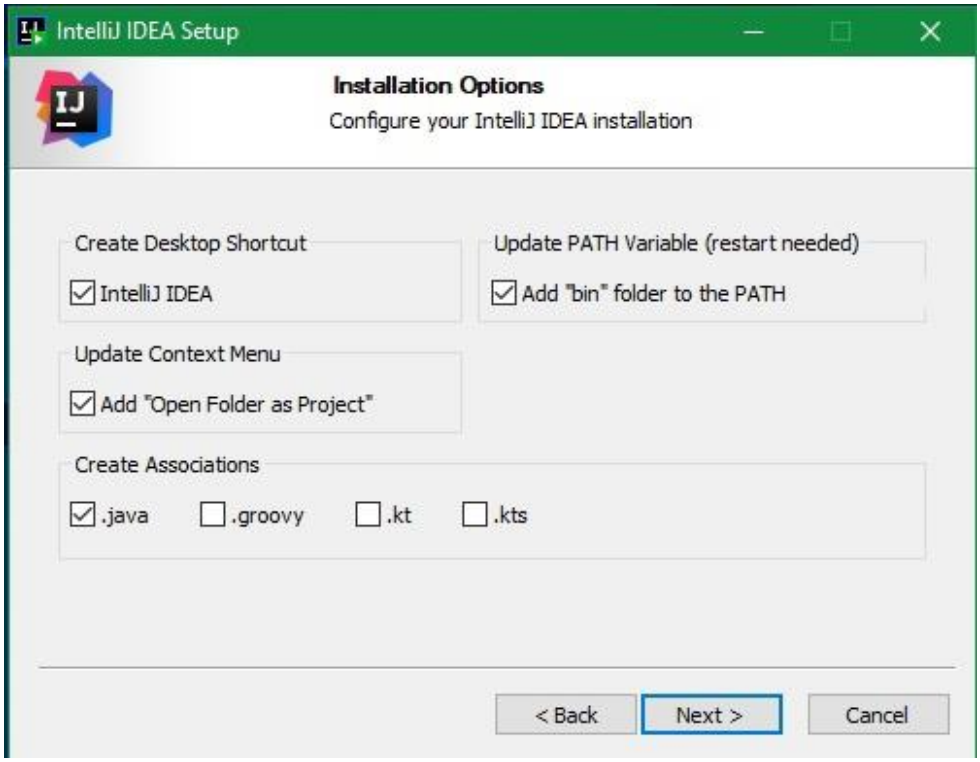

Рисунок 8 - Интерфейс окна найстройки конфигурации установки InteliJ IDEA

Начало установки InteliJ IDEA представлено на рисунке 9.

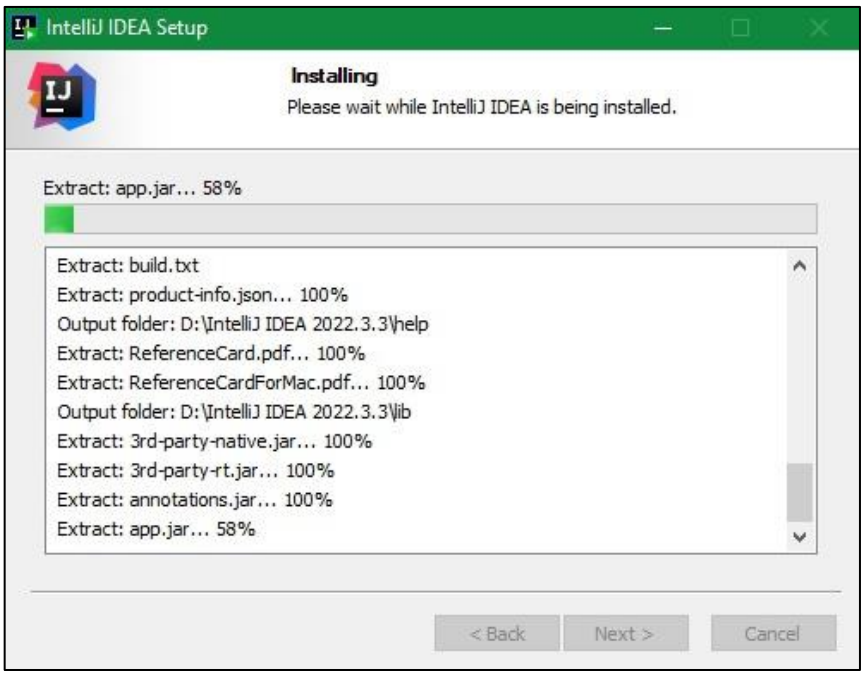

Рисунок 9 - Графический интерфейс окна установки InteliJ IDEA.

Далее идет создание нового проекта, на рисунке 10 представлен графический интерфейс окна приветствия.

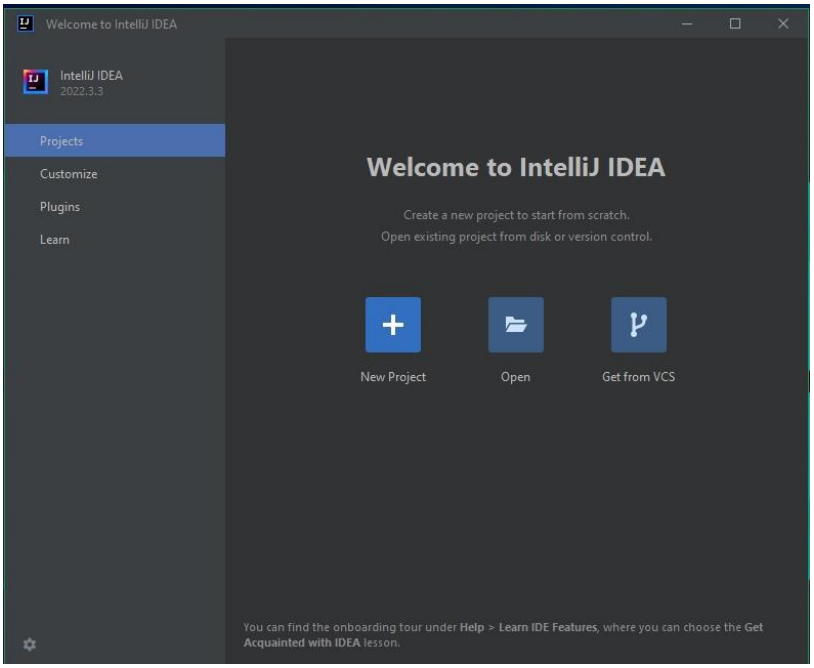

Рисунок 10 - Интерфейс окна приветствия среды разработки InteliJ IDEA

В данном окне необходимо выбрать кнопку новый проект. На рисунке 11 представлено окно создания нового проекта.

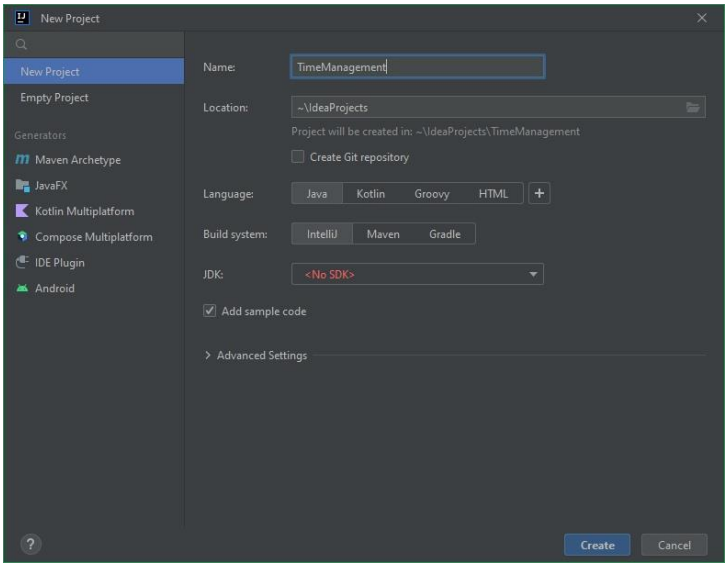

Рисунок 11 - Интерфейс окна создания нового проекта
в интегрированной среде разработки InteliJ IDEA

В поле «Language» необходимо выбрать язык программирование Java, в поле Build system необходимо выбрать «Gradle», в поле выбора «JDK», в появившемся окне, представленном на рисунке 12, было выбрано:

– Version 1.8;

– Vender «Amazon Coretto 11.0.18».

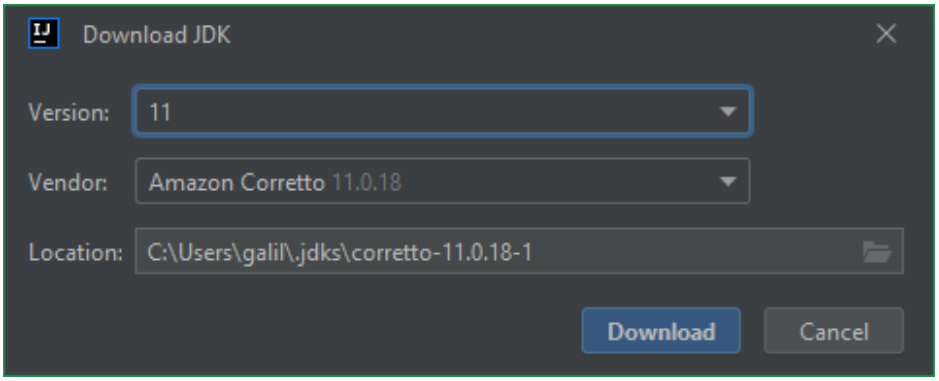

Рисунок 12 - Интерфейс окна выбора JDK

На рисунке 13 представлен интерфейс загрузки пакета JDK.

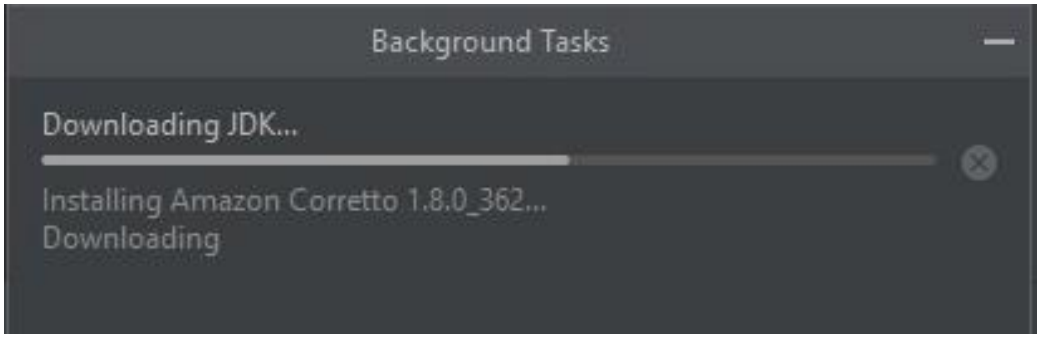

Рисунок 13 - Интерфейс загрузки пакета JDK

После создания проекта необходимо установить Android SDK для разработки приложений на мобильные устройства.

Для установки Android SDK необходимо перейти в окно настроек. Переходим в «Tools», «Android», «SDK Manager». На рисунке 15 представлено окно выбора Android SDK.

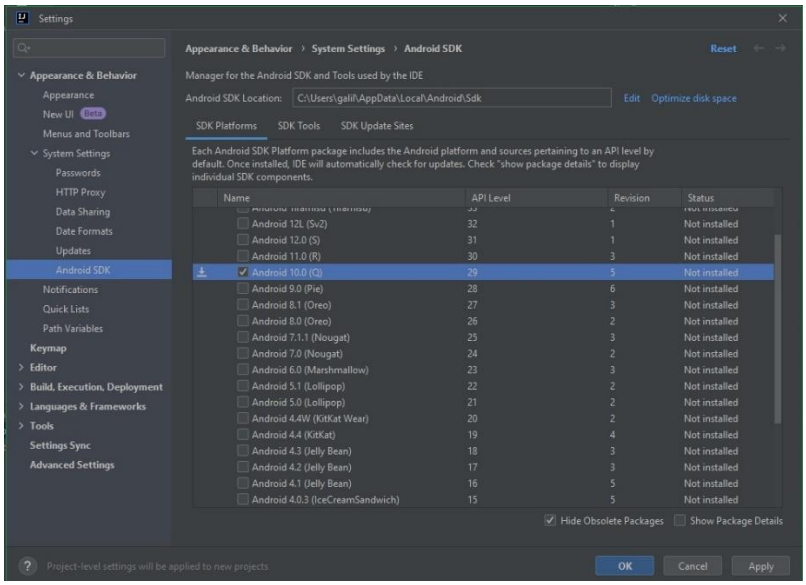

Рисунок 14 - Окно выбора Android SDK

Далее необходимо выбрать Android 10.0 API Level 29 после чего нажать кнопку загрузки. На рисунке 15 представлено окно установки Android SDK.

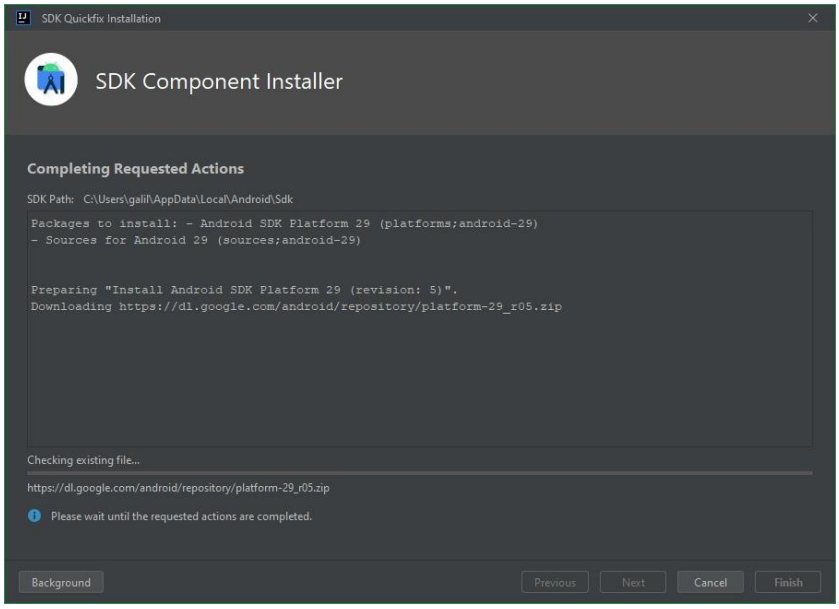

Рисунок 15 - Окно установки Android SDK

На этом этапе интегрированная среда разработки InteliJ IDEA настроена и готова к работе.

Разработка мобильного приложения выполнялась в следующем порядке:

– разработка графического интерфейса с помощью языка разметка XML;
– разметка структуры базы данных;
– разработка программного кода.

До начала разработки графического интерфейса необходимо спроектировать структуру будущего графического интерфейса.

На рисунке 16 представлена структура спроектированного графического интерфейса мобильного приложения.

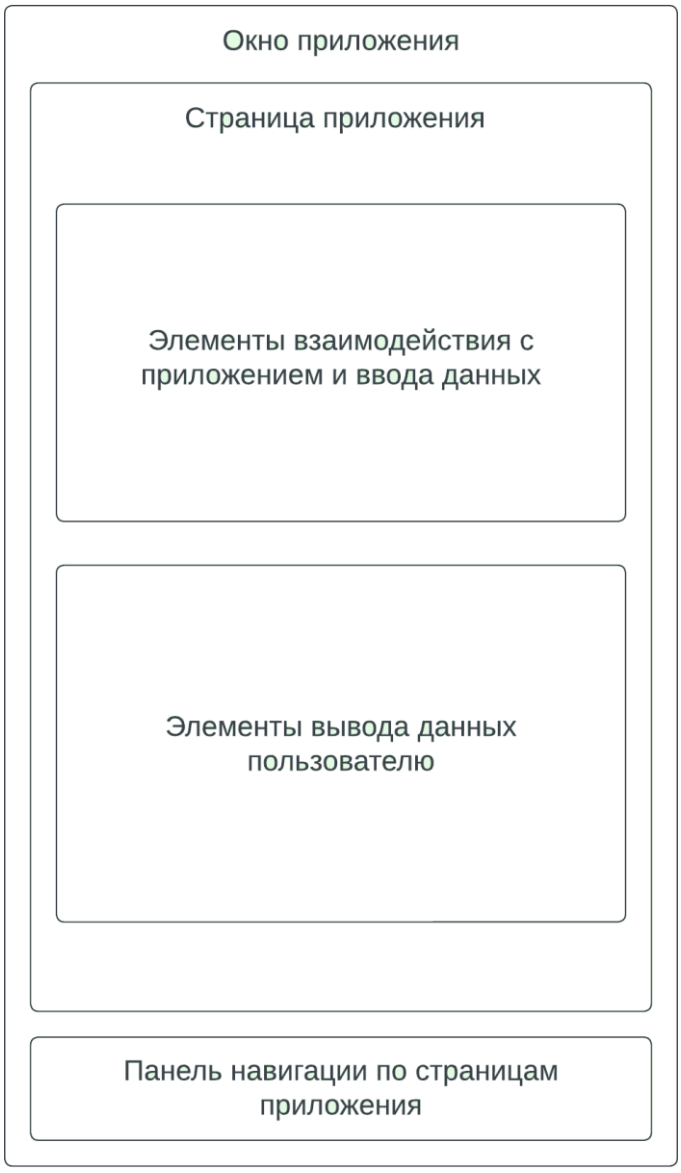

Рисунок 16 - Структура графического интерфейса мобильного приложения

В данной структуре изображены:

– окно приложения - корневой элемент графического интерфейса мобильного приложения, все дальнейшие составные части графического интерфейса являются вложенными в нём;

– страница приложения - является переключаемым окном на котором будут располагаться младшие вложенные элементы;

– элементы взаимодействия с приложением и ввода данных - элемент для ввода данных в приложение, данная структурная единица позволяет пользователю взаимодействовать с приложением, под данной структурной единицей могут подразумеваться - кнопки, поля для ввода;

– элементы вывода данных пользователю - элемент для вывода данных пользователю, представляет из себя - тексты, списки, таблицы;

– панель управления по страницам приложения - предоставляет графический интерфейс для перехода по страницам приложения.

Согласно спроектированной структуре графического интерфейса мобильного приложения, необходимо разметить корневое окно. Файл данного окна будет называться «Activity main». На рисунке 17 представлена разметка корневого окна приложения.

```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".MainActivity">

        <FrameLayout android:layout_width="match_parent"
                     android:layout_height="0dp"
                     android:layout_weight="1"
                     android:background="@color/dark_purple"
                     android:id="@+id/main_content">

        </FrameLayout>

        <com.google.android.material.bottomnavigation.BottomNavigationView
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/nav_bar"
                android:layout_gravity="bottom"
                android:background="@color/darker_purple"
                app:menu="@menu/bottom_nav">
        </com.google.android.material.bottomnavigation.BottomNavigationView>
</LinearLayout>
```

Рисунок 17 - Разметка корневого окна приложения

В разметке корневого окна приложения находится два элемента:

– элемент для отрисовки страниц приложения;

– нижняя панель навигации.

На рисунке 18 представлена разметка страницы «Заказы».

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
             xmlns:app="http://schemas.android.com/apk/res-auto"
             android:layout_width="match_parent"
             android:id="@+id/fragment_orders_layout"
             android:layout_height="match_parent">

    <ExpandableListView android:id="@+id/orders_list"
                        android:layout_width="match_parent"
                        android:layout_height="match_parent"/>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/add_button"
            android:layout_gravity="bottom|end"
            android:layout_marginEnd="16dp"
            android:layout_marginBottom="16dp"
            app:srcCompat="@android:drawable/ic_input_add"/>
</FrameLayout>
```

Рисунок 18 - Разметка страницы «Заказы»

В этой разметке расположены:

– компонент-список для вывода заказов;

– кнопка для регистрации нового заказа.

Была выполнена разметка элемента списка для страницы «Заказы». На рисунках 19 и 20 представлена данная разметка.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:paddingTop="5dp"
              android:paddingStart="50dp"
              android:paddingEnd="10dp"
              android:paddingBottom="5dp"
              android:layout_width="match_parent"
              android:layout_height="match_parent">
    <LinearLayout android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginBottom="5dp"
                  android:orientation="horizontal">

        <TextView android:layout_width="0dp"
                  android:layout_weight="1"
                  android:layout_height="wrap_content"
                  android:textSize="30sp"
                  android:id="@+id/client_output"/>
        <LinearLayout android:layout_width="wrap_content"
                      android:gravity="center_vertical"
                      android:layout_gravity="center_vertical"
                      android:layout_height="wrap_content"
                      android:orientation="vertical">
            <TextView android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:textSize="12sp"
                      android:textColor="@color/light_green"
                      android:id="@+id/if_paid_output"/>
            <LinearLayout android:layout_width="wrap_content"
```

Рисунок 19 - Разметка элемента списка страницы «Заказы»

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:app="http://schemas.android.com/apk/res-auto"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:padding="10dp"
              android:orientation="vertical">
    <TextView android:layout_width="0dp"
              android:layout_height="0dp"
              android:id="@+id/database_id_holder"
              android:visibility="invisible"/>
    <LinearLayout android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:orientation="horizontal">

        <LinearLayout android:layout_width="0dp"
                      android:layout_height="wrap_content"
                      android:layout_weight="1"
                      android:orientation="vertical">
            <TextView android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:text="Заметки"/>
            <TextView android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:id="@+id/notes_output"/>
```

Рисунок 20 - Разметка панели инструментов элемента списка страницы «Заказы»

В первой части находятся элементы «Text View» для вывода информации о заказе, а во второй части инструменты для редактирования заказа.

Для регистрации заказа была размечена страница регистрации заказа, представленная на рисунке 21.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:app="http://schemas.android.com/apk/res-auto"
              xmlns:tools="http://schemas.android.com/tools"
              xmlns:mask="http://schemas.android.com/apk/res-auto"
              android:orientation="vertical"
              android:padding="5dp"
              android:background="@color/dark_purple"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              tools:context=".OrderAddActivity">
    <TextView android:layout_width="0dp"
              android:layout_height="0dp"
              android:id="@+id/id_holder_input"/>
    <TextView android:layout_width="0dp"
              android:layout_height="0dp"
              android:id="@+id/status_id_holder"/>
    <com.google.android.material.textfield.TextInputLayout android:layout_width="match_parent"
                                                           android:layout_height="wrap_content"
                                                           app:hintTextAppearance="@style/Theme.AlGo.InputHintText"
                                                           style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox"
                                                           app:boxBackgroundColor="@color/darker_purple"
                                                           android:id="@+id/client_name_layout"
                                                           android:layout_marginBottom="10dp">

        <com.example.algo.custom.TextInput android:layout_width="match_parent"
                                           android:layout_height="wrap_content"
                                           android:id="@+id/client_name_input"
                                           android:hint="Заказчик"/>
```

Рисунок 21 - Разметка страницы регистрации заказа

На данной разметке:

– поля для ввода информации о заказе;

– кнопка «Сохранить» для регистрации заказа с внесенными данными.

Исходя из поставленной задачи, при просмотре списка заказов имеется необходимость фильтрации по различным параметрам заказа, на основании чего будет размечена страница «Фильтры», представленная на рисунке 22.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              xmlns:app="http://schemas.android.com/apk/res-auto"
              android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:padding="20dp"
              android:background="@color/dark_purple"
              android:orientation="vertical">
    <LinearLayout android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:padding="10dp"
                  android:background="@color/darker_purple"
                  android:orientation="horizontal">
        <Spinner android:layout_width="wrap_content"
                 android:backgroundTint="@color/white"
                 android:layout_height="wrap_content"
                 android:id="@+id/status_filter"/>
    </LinearLayout>


    <LinearLayout android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginTop="15dp"
                  android:padding="5dp"
                  android:orientation="vertical">
        <LinearLayout android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:orientation="horizontal">
```

Рисунок 22 - Разметка страницы «Фильтры»

В разметке находятся:

– поля фильтрации заказов по дате, заказчику, городу и статусу заказов;

– кнопка для применения фильтров.

Поскольку при проектировании поставлена задача формировать статистику, размечена страница «Статистика», представлена на рисунке 23.

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
            xmlns:app="http://schemas.android.com/apk/res-auto"
            android:layout_width="match_parent" android:layout_height="match_parent">
    <LinearLayout android:orientation="vertical"
                  android:padding="10dp"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content">
        <LinearLayout android:layout_width="match_parent"
                      android:layout_height="wrap_content"
                      android:orientation="horizontal"
                      android:padding="10dp" android:layout_marginBottom="10dp">
            <com.google.android.material.floatingactionbutton.FloatingActionButton
                    android:layout_width="wrap_content"
                    android:id="@+id/month_decrease"
                    android:layout_height="wrap_content"
                    app:srcCompat="@drawable/arrow_left"/>
            <TextView android:layout_width="0dp"
                      android:layout_height="wrap_content"
                      android:id="@+id/month_output"
                      android:layout_gravity="center_vertical"
                      android:layout_marginHorizontal="15dp"
                      android:text="Март 2022"
                      android:gravity="center"
                      android:textSize="30sp"
                      android:layout_weight="1"/>
```

Рисунок 23 - Разметка страницы «Статистика»

В данной разметке расположены:

– кнопки для фильтрации статистики по месяцам;

– элементы для вывода статистки;

– поля установки более точной временных отрезков выборки статистики.

На данном этапе разметка графического интерфейса была завершена.

Разработка структуры базы данных включает в себя:

– подключение зависимости необходимых библиотек;

– разметка структуры и связей таблиц;

– написание методов манипуляции с таблицами.

Программный код подключение зависимостей представлен на рисунке 24.

```
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

dependencies {

    implementation 'androidx.appcompat:appcompat:1.4.1'
    implementation 'com.google.android.material:material:1.5.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation "androidx.room:room-runtime:2.4.2"
    annotationProcessor "androidx.room:room-compiler:2.4.2"
}
```

Рисунок 24 - Программный код подключение зависимостей

Подключение зависимостей происходит путем редактирования «dependencies» в файле «Build.gradle», затем необходимо запустить процесс загрузки зависимостей.

Для разработки базы данных необходимо создать файл «AppDatabase.java» нажав на правую кнопку мыши и выбрав New-Java Class.

На рисунке 25 представлено создание файла AppDatabase.java.

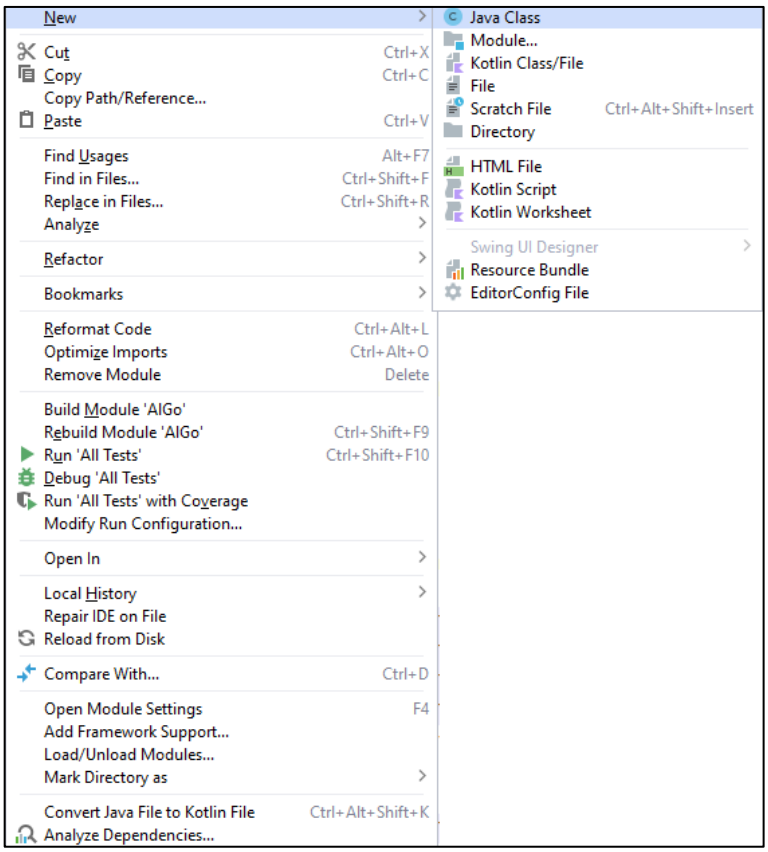

Рисунок 25 - Создание нового файла

В созданном файле необходимо разметить структура базы данных и написать методы манипуляций с таблицами. На рисунке 26, представлен фрагмент программного кода файла «AppDatabase.java».

```
@Database(entities = {Order.class, Status.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    1 usage  1 implementation
    public abstract OrderDao orderDao();
    no usages  1 implementation
    public abstract StatusDao statusDao();

    4 usages
    public static AppDatabase INSTANCE;

    1 usage
    public static AppDatabase getDatabase(Context context) {
        if (INSTANCE == null) {
            synchronized (AppDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE = Room.databaseBuilder(context.getApplicationContext(), AppDatabase.class,  name: "database")
                            .allowMainThreadQueries().addCallback(onCreateCallback)
                            .build();
                }
            }
        }
        return INSTANCE;
    }
```

Рисунок 26 -Фрагмент программного кода файла «AppDatabase.java»

На рисунке 27, представлен программный код создания объекта для работы с базой данных. Инициализация данного объекта реализована с помощью шаблона проектирования Singleton.

```
public static AppDatabase getDatabase(Context context) {
    if (INSTANCE == null) {
        synchronized (AppDatabase.class) {
            if (INSTANCE == null) {
                INSTANCE = Room.databaseBuilder(context.getApplicationContext(), AppDatabase.class, name: "database")
                        .allowMainThreadQueries().addCallback(onCreateCallback)
                        .build();
            }
        }
    }
    return INSTANCE;
}
```

Рисунок 27 -Программный код создания объекта для работы базы данных.

В файле MainActivity будет прописаны слушатели нажатий кнопки, а так же функции включающие активность приложения.

Программный код реализации переходов на страницы «Статистика» и «Заказы», представлен на рисунке 28.

```
    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    if (fragment == ordersFragment) {
        currentFragment = 1;
        ft.hide(statsFragment);
        ft.show(ordersFragment);
    }
    if (fragment == statsFragment) {
        currentFragment = 2;
        ft.hide(ordersFragment);
        ft.show(statsFragment);
    }
    ft.commit();

    invalidateOptionsMenu();
}
```

Рисунок 28 - Реализация перехода на страницу «Статисткиа» и «Заказы»

Программный код, включающий активность приложения, воспроизводятся во время запуска мобильного приложения, представлен на рисунке 29.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    Intent intent = getIntent();
    String ordersSQL = defaultOrdersSQL;
    if (intent.getStringExtra( name: "ordersFilterSQL") != null && !(intent.getStringExtra( name: "ordersFilterSQL")).equals("")) {
        ordersSQL = intent.getStringExtra( name: "ordersFilterSQL");
    }

    actionBar = getSupportActionBar();
    currentFragment = 1;
    BottomNavigationView navigation = (BottomNavigationView) findViewById(R.id.nav_bar);

    navigation.setOnItemSelectedListener(navSelectListener);


    ordersFragment = OrdersFragment.newInstance( activity: this, ordersSQL);
    statsFragment = StatsFragment.newInstance( activity: this);

    FragmentTransaction ft = getSupportFragmentManager().beginTransaction();
    ft.add(R.id.main_content, ordersFragment);
    ft.add(R.id.main_content, statsFragment);
    ft.hide(statsFragment);
    ft.commit();

    actionBar.setTitle("Заказы");
}
```

Рисунок 29 - Фукнции включающие активность приложения

Файл OrderAddActivity отвечает за реализацию регистрирования нового заказа в базе данных.

На рисунках под номером 30 и 31, представлены фрагменты программного кода проверяющий исправность заполнения полей регистрации нового заказа.

```
private boolean is_error() {
    int errors = 0;

    TextInput client_name = (TextInput) findViewById(R.id.client_name_input);
    TextInputLayout client_name_layout = (TextInputLayout) findViewById(R.id.client_name_layout);
    if (shouldShowError(client_name)) {
        errors++;
        showError(client_name_layout);
    } else {
        hideError(client_name_layout);
    }

    TextInput city = (TextInput) findViewById(R.id.city_input);
    TextInputLayout city_layout = (TextInputLayout) findViewById(R.id.city_layout);
    if (shouldShowError(city)) {
        showError(city_layout);
        errors++;
    } else {
        hideError(city_layout);
    }

    TextInput count = (TextInput) findViewById(R.id.count_input);
    TextInputLayout count_layout = (TextInputLayout) findViewById(R.id.count_layout);
    if (shouldShowError(count)) {
        showError(count_layout);
        errors++;
    } else {
        hideError(count_layout);
    }
```

Рисунок 30 - Программный код проверяющий исправность заполнения полей регистрации заказа

```
TextInput sum = (TextInput) findViewById(R.id.sum_input);
TextInputLayout sum_layout = (TextInputLayout) findViewById(R.id.sum_layout);
if (shouldShowError(sum)) {
    showError(sum_layout);
    errors++;
} else {
    hideError(sum_layout);
}

TextInput paid = (TextInput) findViewById(R.id.paid_input);
TextInputLayout paid_layout = (TextInputLayout) findViewById(R.id.paid_layout);
if (shouldShowError(paid)) {
    showError(paid_layout);
    errors++;
} else {
    hideError(paid_layout);
}

TextInput date = (TextInput) findViewById(R.id.date_input);
TextInputLayout date_layout = (TextInputLayout) findViewById(R.id.date_layout);
if (shouldShowError(date)) {
    showError(date_layout);
    errors++;
} else {
    hideError(date_layout);
}
```

Рисунок 31 - Программный код проверяющий исправность заполнения полей регистрации заказа

На рисунке 32, представлен программный код сохранения заказа.

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home:
            finish();
            break;
        case R.id.save_button:
            try {
                if (orderAdd() > 0) {
                    Toast.makeText(getApplicationContext(), text: "Сохранено", Toast.LENGTH_SHORT).show();
                    Intent intent = getIntent();
                    if (intent.getStringExtra( name: "ordersFilterSQL") != null && !(intent.getStringExtra( name: "ordersFilterSQL"))
                        Intent closeIntent = new Intent( packageContext: this, MainActivity.class);
                        closeIntent.putExtra( name: "ordersFilterSQL", intent.getStringExtra( name: "ordersFilterSQL"));
                        startActivity(closeIntent);
                    } else {
                        finish();
                    }
                }
            } catch (ParseException e) {
                Log.e(TAG, e.toString());
            }
    }

    return super.onOptionsItemSelected(item);
}
```

Рисунок 32 -Программный код реализации сохранения заказа

В файле StatsFragment служит для страницы «Статистика», в нем расположены слушатели нажатий всех кнопок, реализация вывода графического интерфейса о статистике для пользователя.

На рисунке 33, представлен программный код инициализации слушателей нажатия кнопок.

```
increase.setOnClickListener(increaseListener);
decrease.setOnClickListener(decreaseListener);
applyButton.setOnClickListener(applyListener);

statsStart.setOnClickListener(dateOpen);
statsStartLay.setOnClickListener(dateOpen);
statsEnd.setOnClickListener(dateOpen);
statsEndLay.setOnClickListener(dateOpen);

orderViewModel = CustomActivity.orderViewModel;

dateSet();
setInitialDateTime();
```

Рисунок 33 - Программный код инициализации слушателей нажатия кнопок

Программный код вывода информации пользователю о статистике представлена на рисунке 34.

```
dateSet();
setInitialDateTime();

orderViewModel.setStats(dateStart.getTimeInMillis(), dateEnd.getTimeInMillis());

orderViewModel.getStatsLiveDate().observe( owner: this, new Observer<Stats>() {
    @Override
    public void onChanged(Stats stats) { updateText(stats); }
});

loadStats();
return view;
```

Рисунок 34 - Программный код вывода инфорации пользователю

Для работы фильтров будет написан программный код в созданном файле FileOrdersActivity.

На рисунке 35 представлена реализация графического интерфейса выбора даты при нажатии на фильтр точного временного отрезка реализованный в методе «dataSate».

```
public void loadStats() {
    orderViewModel.setStats(dateStart.getTimeInMillis(), dateEnd.getTimeInMillis());

    orderViewModel.getStatsLiveDate().observe( owner: this, new Observer<Stats>() {
        @Override
        public void onChanged(Stats stats) { updateText(stats); }
    });
}

2 usages
private void updateText(Stats stats) {
    DecimalFormat formatter = new DecimalFormat( pattern: "###,###");
    moneySumOutput.setText(formatter.format(stats.moneySum).replaceAll( regex: ",", replacement: " "));
    productsSumOutput.setText(formatter.format(stats.productsSum).replaceAll( regex: ",", replacement: " "));
    orderCountOutput.setText(formatter.format(stats.ordersCount).replaceAll( regex: ",", replacement: " "));
}
```

Рисунок 35 - Реализация графического интерфейса выбора даты

После выбора даты происходит форматирование даты, программный код форматирования выбранной даты представлен на рисунке 36.

```
private void setInitialDateTime() {
    dateSet();
    SimpleDateFormat dateFormatter = new SimpleDateFormat( pattern: "dd.MM.yyyy");
    dateStartInput.setText(dateFormatter.format(dateStart.getTime()));
    dateEndInput.setText(dateFormatter.format(dateEnd.getTime()));
}
```

Рисунок 36 - Програмный код формитирования даты

После форматирования даты, необходимо найти заказы на выбранные даты в базе данных с последующим выводом пользователю.

Для этого отправляется SQL запрос поиска заказов на выбранный промежуток времени, после чего выбранные заказы отображаются пользователю, данный процесс представлен на рисунках 37 и 38.

```
queryString += "SELECT `order`.*, `status`.name, `status`.color FROM `order` JOIN `status` ON `order`.status_id = `status`.id";
```

Рисунок 37 - Программыний код SQL запрос поиска заказов на выбранные даты

```
@Override
public boolean onOptionsItemSelected(@NonNull MenuItem item) {
    switch (item.getItemId()) {
        case R.id.filter_orders:
            Intent intentFilter = getIntent();
            String ordersSQL = defaultOrdersSQL;
            if (intentFilter.getStringExtra( name: "ordersFilterSQL") != null && !(intentFilter.getStringExtra
                ordersSQL = intentFilter.getStringExtra( name: "ordersFilterSQL");
            }

            Intent intent = new Intent( packageContext: this, FilterOrdersActivity.class);
            intent.putExtra( name: "ordersFilterSQL", ordersSQL);
            startActivity(intent);
    }
    return super.onOptionsItemSelected(item);
}
```

Рисунок 38 - Программный код вывода заказов по фильтру выборы даты

В файле под названием OrderFragment реализован элемент «Заказ», на данном файле реализованы инструменты редактирования заказа, их удаление, и присваивание статуса. На рисунке 39 представлен фрагмент программного кода данного файла.

```
public OrdersFragment(Activity mActivity, String SQL){
    activity = mActivity;
    this.SQL = SQL;
}

1 usage
public static OrdersFragment newInstance(Activity activity, String SQL) {
    return new OrdersFragment(activity, SQL);
}
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_orders, container,  attachToRoot: false);
    ExpandableListView listView = (ExpandableListView) view.findViewById(R.id.orders_list);
    OrderViewModel orderViewModel = CustomActivity.orderViewModel;

    orderViewModel.setOrders(SQL);
    ArrayList<OrderStatus> orders = orderViewModel.getOrders();

    OrdersListAdapter ordersListAdapter = new OrdersListAdapter(getContext(), orders, orderViewModel,
            getActivity(), new EditListener(), listView);

    orderViewModel.getLiveDataOrders().observe( owner: this, new Observer<Map<Order, Status>>() {
        @Override
        public void onChanged(Map<Order, Status> orderStatusMap) {
            ordersListAdapter.setOrders(orderViewModel.getOrders());
        }
    });
```

Рисунок 39 - Фрагмент программного кода элемента «Заказ»

В файле OrderListAdapter реализован вывод элемента «Заказ» на страницу «Заказы», на рисунке 40 показан фрагмент кода данного файла.

```
View ringView = (View) convertView.findViewById(R.id.status_output_ring);
switch ((int) orders.get(orderPosition).status_id) {
    case 1:
        ringView.setBackgroundResource(R.drawable.red_ring);
        break;
    case 2:
        ringView.setBackgroundResource(R.drawable.yellow_ring);
        break;
    case 3:
        ringView.setBackgroundResource(R.drawable.green_ring);
        break;
}
```

Рисунок 40 - Реализация вывода элемента заказ на страницу «Заказы»

2.2.1.1 Настройка взаимодействия с базой данных

Для взаимодействия мобильного приложения с базой данных SQLite необходимо произвести следующие действия:

– создать разметку базы данных;

– реализовать регистрирование нового заказа в базу данных;

– реализовать считывание информации с базы данных.

Разметка базы данных реализуется в файле под названием «AppDatabase.java». На рисунке 41 представлен программный код разметки базы данных.

```
@Override
public void onCreate(@NonNull SupportSQLiteDatabase db) {
    ContentValues contentValues = new ContentValues( size: 3);

    contentValues.put("id", 1);
    contentValues.put("name", "Принято");
    contentValues.put("color", "#7B001C");
    db.insert( table: "status", CONFLICT_REPLACE, contentValues);

    contentValues.clear();
    contentValues.put("id", 2);
    contentValues.put("name", "Отправлено");
    contentValues.put("color", "#FFCF40");
    db.insert( table: "status", CONFLICT_REPLACE, contentValues);

    contentValues.clear();
    contentValues.put("id", 3);
    contentValues.put("name", "Доставлено");
    contentValues.put("color", "#004524");
    db.insert( table: "status", CONFLICT_REPLACE, contentValues);

    super.onCreate(db);
}
```

Рисунок 41 - програмный код разметки базы данных

Для регистрации заказа в базе данных необходимо отправить запрос на регистрацию в базу данных.На рисунке 42 представлен программный код обращение к базе данных.

```
1 usage   1 implementation
@Insert(onConflict = REPLACE)
long insertOrder(Order order);
```

Рисунок 42 - программный код обращение к базе данных

Считывание информации с базы данных необходимо для вывода заказов пользователю, реализацию фильтров и реактивного обновления данных. На рисунке 43 представлен программный код считывания информации с базы данных.

```
1 usage
@SuppressLint("RestrictedApi")
private void setLiveData(String stringSQL) {
    SimpleSQLiteQuery SQL = new SimpleSQLiteQuery(stringSQL);
    orders = db.getInvalidationTracker().createLiveData(new String[]{"order","status"}, inTransaction: false, ()->{
        return orderDao.getAllOrders(SQL);
    });
}

1 usage
@SuppressLint("RestrictedApi")
private void setSumAndCountLiveData(long dateStart, long dateEnd) {
    statsLiveData = db.getInvalidationTracker().createLiveData(new String[]{"order","status"}, inTransaction: false, ()->{
        return orderDao.getStats(dateStart, dateEnd);
    });
}
```

Рисунок 43 - Программный код считывания информации с базы данных.

2.2.2 Интерфейс разработанного мобильного приложения.

Графический интерфейс пользователя - это способ взаимодействия пользователя с компьютерной программой или операционной системой, использующий графические элементы, такие как кнопки, меню, окна и диалоговые окна. Графический интерфейс делает работу с компьютером более интуитивной и удобной для пользователей, которые не обладают знаниями работы с персональным компьютером [11].

Графический интерфейс обычно состоит из следующих элементов:

– окна - это прямоугольные области на экране, которые содержат информацию или предоставляют доступ к функциям программы;

– кнопки - это графические элементы, которые позволяют пользователю выполнить определенное действие, например, открыть файл или сохранить изменения;

– меню - это список команд, которые пользователь может выбрать для выполнения определенных действий;

– диалоговые окна - это окна, которые появляются при выполнении определенных действий и запрашивают у пользователя дополнительную информацию или подтверждение действия;

На рисунках 44, 45 представлен графический интерфейс страниц «Заказы», «Статитсика».

Рисунок 44 - Графический интерфейс страницы «Заказы»

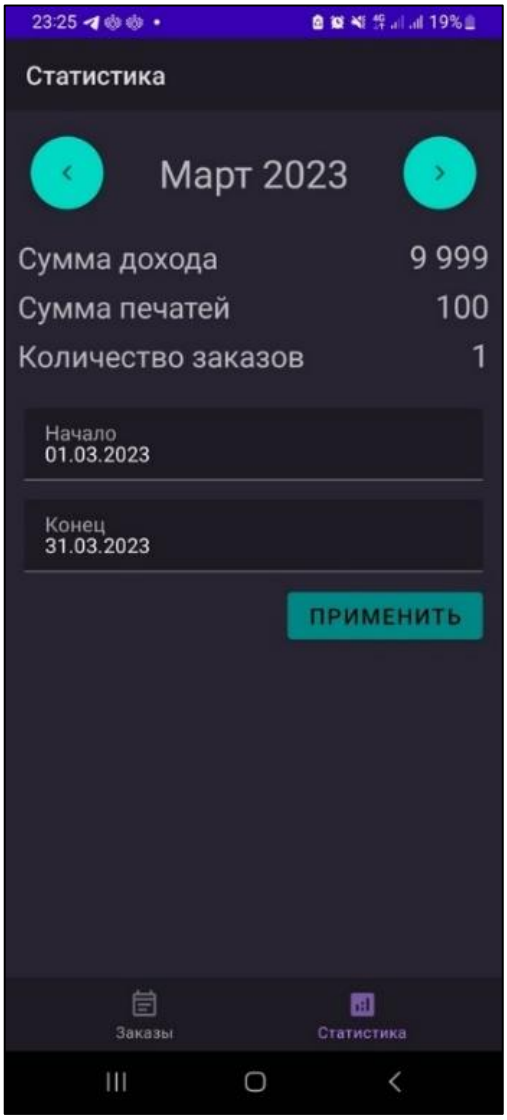

Рисунок 45 - Графический интерфейс страницы «Статистика»

2.2.3Разработка руководства пользователя

Для того чтобы пользоваться мобильным приложением «Time management» необходимо запустить приложение, откроется вкладка с заказами, чтобы добавить заказ необходимо нажать кнопку «+» в правом нижнем углу приложения. После нажатия на кнопку откроется вкладка регистрации заказа. На рисунке 45 показана интерфейс страницы для регистрации заказа.

После ввода всех данных необходимо нажать «СОХРАНИТЬ» в правом верхнем углу приложения. После сохранения заказа, на странице «Заказы», появится созданный заказ. На рисунке 46, продемонстрирован интерфейс страницы «Заказы» после добавления одного заказа.

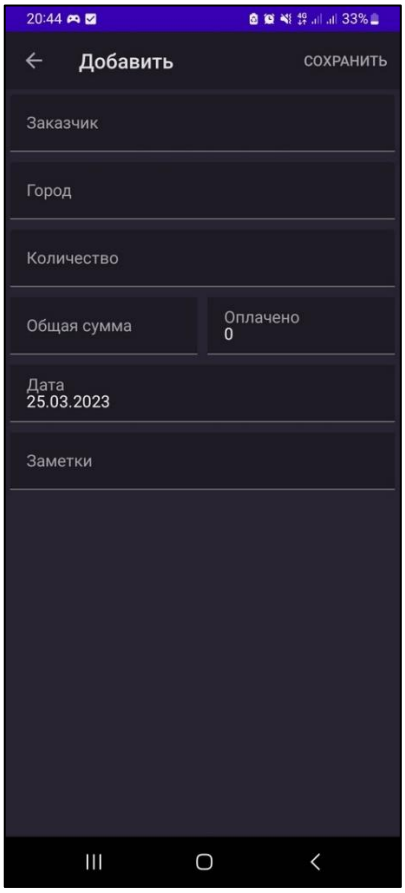

Рисунок 46 - Интерфейс страницы для регистрации заказа

Рисунок 47 -Интерфейс страницы «Заказы» после добавения заказа

Для того чтобы редактировать существующий заказ необходимо нажать на заказ, после чего выйдут дополнительные инструменты для изменения заказа, и присваивания статуса для заказа. На рисунке 47, представлены инструменты манипуляции с заказом.

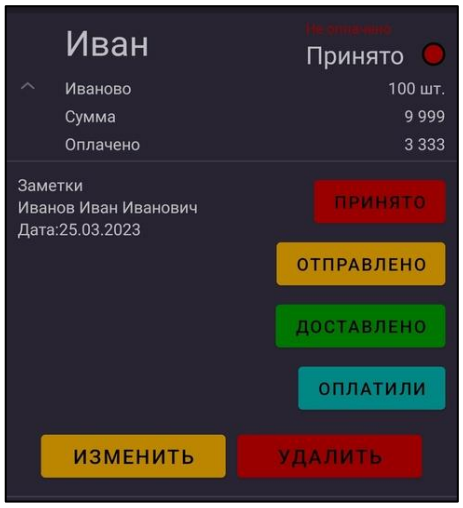

Рисунок 48 -Инструменты манипуляции с заказами

Для просмотра статистики по существующим заказам необходимо нажать кнопку «Статистика». На рисунке 48, представлен графический интерфейс страницы «Статистика».

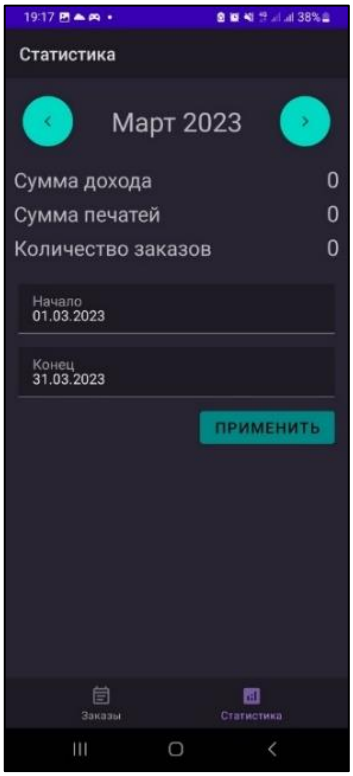

Рисунок 49 - Графический интерфейс страницы «Статистика»

Для использования фильтров по месяцам необходимо нажимать кнопки направлений расположенных в верхней части экрана во вкладке статистика. Для применения более точных временных отрезков необходимо нажать даты с названием «Начало» и выбрать «Конец» даты, при нажатии будет предоставляться выбор даты в виде календаря. На рисунке 50, представлен интерфейс календаря выбора точного временного отрезка для фильтрации.

Рисунок 50 - Интерфейс календаря выбора точного временного отрезка для фильтрации

ЗАКЛЮЧЕНИЕ

Мобильное приложение - или приложение для мобильных устройств, также может встречаться под названиями сетевое приложение, онлайн или офлайн приложение, приложение для смартфона - программное изделие, разновидность прикладного программного обеспечения, предназначенная для работы на смартфонах, планшетах и других мобильных, портативных, переносных, карманных устройствах. Обеспечивает без привязки к стационарному компьютеру, необходимые пользователю взаимодействия со Всемирной Сетью, либо устанавливается на устройстве благодаря такому взаимодействию - после скачивания на носимое пользователем устройство.

Целью дипломной работы является разработка мобильного приложения «Time management» для учета заказов и формирования статики.

Произведен анализ предметной области, в ходе которого рассмотрен бизнес-процесс обработки заказов.

Анализ показал, что указанный бизнес-процесс слабо автоматизирован и может быть улучшен посредством внедрения мобильного приложения, автоматизирующей функции учета заказов и формирования статистики.

Исследование готовых программных продуктов аналогичного назначения позволило сделать вывод о целесообразности собственной разработки, для чего был произведен выбор средств реализации мобильного приложения, а также выделены основные достоинства и недостатки исследованных, проанализированных мобильных приложений.

Для разработки мобильного приложения «Time management» использовались следующие программные средства:

- Amazon coretto 11;
- Android SDK API 29;
- СУБД SQLite;
- Room для работы с СУБД;
- язык программирования Java;
- интегрированная среда разработки InteliJ IDEA.

Интегрированная среда разработки InteliJ IDEA была использована для написания программного кода. СУБД SQLite это быстрая и легкая встраиваемая однофайловая СУБД, которая не имеет сервера и позволяет хранить всю базу локально на одном устройстве, использовалась в качестве базы данных. Для работы SQLite не нужны сторонние библиотеки или службы..Room использовалась для работы с базой данных SQLite. Android SDK API 29 – это дополнительный набор инструментов, которые помогают написать код, запустить тестирование и отладку, проверить работу приложения на различных версиях операционной системы и оценить результат в реальном времени. Amazon coretto 11 - это бесплатная многоплатформенная версия пакета средств разработки Open Java, готовая к использованию в рабочей среде. Corretto поставляется с долгосрочной поддержкой, которая включает в себя повышение производительности и исправления безопасности. Язык программирования Java - строго типизированный объектно-ориентированный язык программирования общего назначения, использовался в качестве основного языка программирования, на данном языке программирования были реализованы основные функции мобильного приложения [9].

В ходе выполнения дипломной работы были продемонстрированы теоретические и практические навыки и умения, полученные в рамках обучения. Подводя ее итоги можно выделить следующие наиболее значимые достигнутые результаты:

– произведен анализ предметной области в сфере мобильных приложений для учета заказов и формирования статистки.

– приобретены навыки программирования на языке Java;

– приобретены навыки работы с базой данных SQLite;

– приобретен навык работы с InteliJ IDEA;

– спроектированы: база данных, мобильное приложение для учета заказов и формирования статистки;

– разработана информационная система;

– составлена спецификация и проведено обоснование функциональных и нефункциональных требований к системе учета заказов;

– разработка модели программного продукта, включающая в себя: разработку блок-схем алгоритма работы мобильного приложения «Time management», возможные сценарии взаимодействия с ней пользователя показанных в главе посвященной аналитической части работы;

– разработано руководство пользователя для использования мобильного приложения «Time management»;

При написании данной работы были продемонстрированы результаты разработки программы в виде изображений готового интерфейса программы, структуры ее базы данных, а также фрагментов кода, представленных как примеры.

При решении задачи были изучены теоретически методы предъявления и анализа требований к программному обеспечению, пользовательскому интерфейсу.

Было разработано приложение, которое соответствует предъявленным требованиями дипломной работой, и функционирует согласно описанным требованиям. Также были закреплены и получены новые навыки работы в среде разработки InteliJ IDEA [8].

Исходя из технической характеристики проекта, можно сделать вывод, что разработанное мобильное приложение является современной и выполняет поставленные задачи без длительных задержек и ошибок.

Пользователям не следует заниматься занесением информации о заказах на бумажные носители, либо пользоваться электронными таблицами, также пользователю нет необходимости вручную вести статистку о заказах, что раньше могло занимать физическое пространство и время [5].

СПИСОК ЛИТЕРАТУРЫ

1 ГОСТ 19.102-77. Стадии разработки. - Введ. 01.01.1980. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2010. - 2 с. - (Единая система программной документации).

2 ГОСТ 19.201-78. Техническое задание. Требования к содержанию и оформлению. - Введ. 01.01.1980. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2013. - 2 с. - (Единая система программной документации).

3 ГОСТ 19.402-78. Описание программы. Введ. 01.01.1980. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2013. - 2 с. - (Единая система программной документации).

4 ГОСТ 19.404-79. Пояснительная записка. Требования к содержанию и оформлению. - Введ. 01.01.1981. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2013. - 2 с. - (Единая система программной документации).

5 ГОСТ 19.504-79. Руководство программиста. Требования к содержанию и оформлению. - Введ. 01.01.1980. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2011. - 1 с. - (Единая система программной документации).

6 ГОСТ 19.505-79. Руководство оператора. Требования к содержанию и оформлению. - Введ. 01.01.1980. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2014. - 2 с. - (Единая система программной документации).

7 ГОСТ 19.701-90. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. - Взамен ГОСТ 19.002-80, ГОСТ 19.003-80; введ. 01.01.1992. - М.: Межгосударственный совет по стандартизации, метрологии и сертификации, 2011. - 23 с. - (Единая система программной документации).

8 ГОСТ 2.105-95. Общие требования к текстовым документам. - Взамен ГОСТ 2.105-79, ГОСТ 2.906-71; введ. 01.07.1996. - Минск: Межгосударственный совет по стандартизации, метрологии и сертификации, 2011. - 19 с. - (Межгосударственный стандарт. Единая система конструкторской документации).

9 ГОСТ 7.1. – 2014 «Библиографическая запись. Библиографическое описание».

10 ГОСТ 7.32. – 2012 «Система стандартов по информации,библиотечному и издательскому делу «Отчет о научно-исследовательской работе».

11 ГОСТ 7.82. – 2010 «Библиографическая запись».

12 Блох, Д. Java Эффективное программирование / Д. Блох. - М.: Лори, 2016. - 440 с.

13 Васильев, А. Java. Объектно-ориентированное программирование: Учебное пособие Стандарт третьего поколения / А. Васильев. - СПб.: Питер, 2013. - 400 с.

14 Давыдов, С. IntelliJ IDEA. Профессиональное программирование на Java / С. Давыдов. - СПб.: BHV, 2005. - 800 с.

15 Дэрси, Л. Разработка приложений для Android-устройств. Т. 1: Базовые принципы / Л. Дэрси, Ш. Кондер. - М.: Лори, 2014. - 402 с.

16 Кожанова Е. Печати и штампы в организации [Электронный ресурс] Режим доступа: URL: https://delo-press.ru/journals/documents/sovremennoe-deloproizvodstvo/42588-pechati-i-shtampy-v-organizatsii/ (дата обращения: 19.12.2022).

17 Кристофер, Х. Штампы и печати в изготовлении украшений / В.Д. Колдаев; Под ред. Л.Г. Гагарина. - М.: ИД Форум, Инфра-М, 2012. - 416 с.

18 Лавровская, О.Б. Технические средства информатизации. Практикум: Учебное пособие для студ. учреждений сред. проф. образования / О.Б. Лавровская. - М.: ИЦ Академия, 2012. - 208 с.

19 МакГрат, М. Программирование на Java для начинающих / М. МакГрат. - М.: Эксмо, 2016. - 192 с.

20 Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. / Р. Мартин. – СПб : Питер, 2021. – 352 с.

21 Назаров, С.В. Архитектура и проектирование программных систем: Монография / С.В. Назаров. - М.: НИЦ Инфра-М, 2013. - 351 с.

22 Нимейер, П. Программирование на Java / П. Нимейер, Д. Леук. - М.: Эксмо, 2018. - 448 с.

23 Парфилова, Н.И. Программирование: Основы алгоритмизации и программирования: Учебник / Н.И. Парфилова; Под ред. Трусова Б.Г. - М.: Academia, 2018. - 32 с.

24 Ричард, Б. Технологии печати / А.Н. Васильев. - М.: Эксмо, 2014. - 416 с.

25 Рудикова, Л.В. Базы данных. Разработка приложений / Л.В. Рудикова. - СПб.: BHV, 2006. - 496 с.

26 Рылько, М. Компьютерные технологии в проектировании / М. Рылько. - М.: АСВ, 2022. - 326 с.

27 Семакин, И.Г. Основы алгоритмизации и программирования: Учебник / И.Г. Семакин. - М.: Academia, 2017. - 328 с.

28 Советов, Б.Я. Архитектура информационных систем: Учебник / Б.Я. Советов. - М.: Академия, 2008. - 240 с.

29 Соловьев, И.В. Проектирование информационных систем. Фундаментальный курс / И.В. Соловьев, А.А. Майоров. - М.: Академический проект, 2019. - 398 с.

30 Черноруцкий, И.Г. Методы оптимизации. Компьютерные технологии / И.Г. Черноруцкий. - СПб.: BHV, 2021. - 384 с.

31 Шпак, Ю.А. Проектирование баз данных. Просто как дважды два / Ю.А. Шпак. - М.: Эксмо, 2018. - 304 с.

32 Щитов, И.Н. Введение в методы оптимизации. / И.Н. Щитов. - М.: Высшая школа, 2018. - 206 с.

Приложение А
Техническое задание на программный продукт

Министерство образования и науки Республики Башкортостан
Государственное бюджетное профессиональное образовательное учреждение
Уфимский государственный колледж технологии и дизайна

Рассмотрено на заседании
ЦК математических и
естественнонаучных дисциплин
Протокол № ____
«_____» _____ 202__ г
Председатель ЦК
_____О.Н. Заглядина

ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА
ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

Ф. И. О. студента Галилов Георгий Владимирович

Учебная группа И-19-19

Специальность Информационные системы (по отраслям)

Тема: Разработка мобильного приложения «Time management»

1. ВВЕДЕНИЕ

Настоящее техническое задание распространяется на разработку мобильного приложения «Time managament».

Мобильное приложение позволит вести учет заказов и формировать статистику по зарегистрированным заказам.

2. ОСНОВАНИЕ ДЛЯ РАЗРАБОТКИ

Мобильное приложение разрабатывается на основании приказа колледжа «Об утверждении тем, руководителей и консультантов дипломных работ студентов» № 2/395а от 05.12.2022 г.

3. НАЗНАЧЕНИЕ

Разработка производится с целью создания мобильного приложения,с помощью которого можно вести учет заказов и формировать статистику.

4. ТРЕБОВАНИЯ К ПРОГРАММНОМУ ПРОДУКТУ

4.1 Требования к функциональным характеристикам

Добавление и редактирование данных:

- заказчик;
- город;
- количество;
- общая сумма;
- оплачено;
- дата;
- заметки.

С возможностями: сортировки зарегистрированных заказов по дате, заказчику, городу, статусу. Отображению внесеной информации по заказам. Изменение уже зарегистрированного заказа, возможность присваивания статуса, выведение статистики на основе выбранных фильтров

4.2 Требования к надежности

Действия пользователя не должны приводить к некорректной или неправильной работе мобильного приложения.

4.3 Требования к составу и параметрам технических средств

Требования к составу и параметрам технических средств накладываются к требованиям мобильного устройства.

4.4 Требования к программной совместимости

Программа должна быть запущена в операционной среде Windows 10/11 / Linux Ubuntu/Fedora/Debian/Red OS.

Мобильное приложение должно быть запущено на мобильном устройстве с операционной системой Android.

5. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Вместе с программным продуктом должны предоставляться пояснительная записка и руководство пользователя. Пособие должно содержать всю необходимую информацию по эксплуатации в отдельном разделе.

6. ЭТАПЫ РАЗРАБОТКИ

№	Название этапа	Срок	Отчетность
1	Разработка ТЗ	До 06.03.2023	Техническое задание
2	Разработка мобильного приложения	До 30.04.2023	Код программы
3	Тестирование, внедрение	До 23.05.2023	Программный продукт

Дата выдачи задания: «02» декабря 2022 г.

Руководитель: Валеева Г.Р.

Задание принял к исполнению: Галилов Г.В.

ПРИЛОЖЕНИЕ В
Фрагменты программного кода мобильного приложения

```
package com.example.algo;

import static android.database.sqlite.SQLiteDatabase.CONFLICT_REPLACE;
import static com.example.algo.App.onCreateCallback;

@Database(entities = {Order.class, Status.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    public abstract OrderDao orderDao();
    public abstract StatusDao statusDao();

    public static AppDatabase INSTANCE;

    public static AppDatabase getDatabase(Context context) {
        if (INSTANCE == null) {
            synchronized (AppDatabase.class) {
                if (INSTANCE == null) {
                    INSTANCE = Room.databaseBuilder(context.getApplicationCon-
text(), AppDatabase.class, "database")
                            .allowMainThreadQueries().addCallback(onCreate-
Callback)
                            .build();
                }
            }
        }
        return INSTANCE;
    }

    public static RoomDatabase.Callback onCreateCallback = new RoomData-
base.Callback() {

        @Override
        public void onCreate(@NonNull SupportSQLiteDatabase db) {
            ContentValues = new ContentValues(3);

            contentValues.put("id", 1);
            contentValues.put("name", "Принято");
            contentValues.put("color", "#7B001C");
            db.insert("status", CONFLICT_REPLACE, contentValues);

            contentValues.clear();
            contentValues.put("id", 2);
            contentValues.put("name", "Отправлено");
            contentValues.put("color", "#FFCF40");
            db.insert("status", CONFLICT_REPLACE, contentValues);

            contentValues.clear();
            contentValues.put("id", 3);
            contentValues.put("name", "Доставлено");
            contentValues.put("color", "#004524");
            db.insert("status", CONFLICT_REPLACE, contentValues);

            super.onCreate(db);
        }
    };
}


package com.example.algo;

import android.app.DatePickerDialog;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.DatePicker;
import android.widget.Spinner;
import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.ActionBar;
import androidx.lifecycle.ViewModelProvider;
import androidx.sqlite.db.SimpleSQLiteQuery;
import com.example.algo.custom.CustomActivity;
import com.example.algo.custom.TextInput;
import com.example.algo.models.Order;
import com.example.algo.models.OrderStatus;
import com.example.algo.models.OrderViewModel;
import com.example.algo.models.Status;
import org.w3c.dom.Text;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.List;
import java.util.Map;

public class FilterOrdersActivity extends CustomActivity {
    Calendar dateStart = Calendar.getInstance();
    Calendar dateEnd = Calendar.getInstance();
    TextInput dateStartInput;
    TextInput dateEndInput;

    ActionBar actionBar;
    Spinner spinner;

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.orders_filter);

        actionBar = getSupportActionBar();
        assert actionBar != null;
        actionBar.setDisplayHomeAsUpEnabled(true);
        actionBar.setDisplayShowHomeEnabled(true);
        actionBar.setTitle("Фильтры");


        spinner = findViewById(R.id.status_filter);
        dateStartInput = findViewById(R.id.date_start_input);
        dateEndInput = findViewById(R.id.date_end_input);

        ArrayAdapter<?> spinnerAdapter = ArrayAdapter.createFromResource(this,
R.array.status_spinner,
                android.R.layout.simple_spinner_item);
        spinnerAdapter.setDropDownViewResource(R.layout.spinner_item);
```