

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет **Прикладной информатики**

Направление подготовки **09.03.03 Прикладная информатика**

Образовательная программа **Мобильные и сетевые технологии**

КУРСОВОЙ ПРОЕКТ

Тема: «Разработка мобильного приложения на iOS для учета трат»

Обучающийся: Авдиенко Данила Андреевич, К3140

Санкт-Петербург 2024

СОДЕРЖАНИЕ

СОДЕРЖАНИЕ.....	2
1 ВВЕДЕНИЕ.....	3
1.1 Актуальность темы	3
1.2 Цель проекта.....	4
1.3 Задачи проекта	4
2 ОСНОВНАЯ ЧАСТЬ.....	5
2.1 Суть проекта	5
2.2 Описание работы над проектом	5
2.3 Мои задачи	7
2.4 Анализ работы	11
2.4.1 Что получилось.....	11
2.4.2 Основные трудности.....	11
2.4.3 Что не получилось	12
2.4.4 Чему я научился за время проекта.....	12
2.5 Взаимодействие с командой	13
2.6 Взаимодействие с руководителем проекта	13
2.7 Оценка работы руководителя	13
3 ЗАКЛЮЧЕНИЕ.....	14
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	15
ПРИЛОЖЕНИЕ.....	16

1 ВВЕДЕНИЕ

1.1 Актуальность темы

Управление личными финансами является одной из важнейших задач для современного человека, независимо от уровня дохода или профессии. Грамотный контроль расходов и доходов позволяет не только избежать хаотичных трат, но и сформировать стратегию для планирования бюджета, позволяющую достичь долгосрочных финансовых целей. В то же время неосведомлённость о собственных расходах зачастую приводит к накоплению долгов, снижению качества жизни и растрате сбережений.

Вследствие этого значительно возрос интерес к цифровым инструментам, упрощающим процесс планирования бюджета и повышающим уровень финансовой осведомленности. Широкое распространение мобильных устройств и доступность интернета способствуют тому, что многие люди предпочитают решать финансовые задачи «на ходу», используя удобные приложения, способные автоматически анализировать транзакции и представлять их пользователю в удобном виде. При этом актуальным остаётся вопрос конфиденциальности данных и их корректного учёта, что делает создание безопасных и надёжных приложений особенно важным.

Помимо повседневного удобства, цифровые сервисы для учёта финансов способствуют повышению финансовой грамотности. Возможность наглядно увидеть структуру своих доходов и расходов, а также отследить динамику изменения трат стимулирует пользователей принимать более взвешенные решения, корректировать бюджетные планы и осознанно подходить к своим тратам. Благодаря такому отслеживанию люди могут проще адаптироваться к изменениям в бюджете, не выходя за рамки заданных финансовых лимитов.

Именно поэтому рынок мобильных приложений насыщен самыми разными финансовыми инструментами, что отражает востребованность подобных решений со стороны пользователей. Актуальность разработки

нового приложения во многом определяется стремлением адаптировать функционал к специфическим потребностям определённой аудитории или улучшить уже существующие возможности. Таким образом, тема учёта и анализа личных финансов не теряет своей значимости, а развитие мобильных технологий позволяет обернуть «финансовый ежедневник» в удобную программу, доступную где угодно.

Всё это подчёркивает актуальность темы разработки мобильного приложения на iOS для учета трат, которое будет не просто хранить доходы и расходы, но и позволять пользователю глубже понимать свою финансовую ситуацию, ставить конкретные цели и стремиться к их достижению.

1.2 Цель проекта

Цель проекта – Создать мобильное приложение на iOS, которое позволяет пользователям удобно сохранять и анализировать свои траты.

1.3 Задачи проекта

При реализации проекта перед нашей командой стояло множество задач, которые можно распределить на два основных блока: дизайн и разработка.

За дизайн отвечали два члена команды, они выполняли следующие пункты:

- Подбор цветовой гаммы
- Проектирование всех экранов мобильного приложения в инструменте Figma

- Создание дизайна для конкретных элементов приложения

За разработку отвечали три члена команды, ими были выполнены следующие пункты:

- Проектирование архитектуры мобильного приложения
- Верстка всех экранов мобильного приложения
- Подключение приложения к стороннему инструменту Firebase
- Настройка взаимодействия приложения и Firebase

2 ОСНОВНАЯ ЧАСТЬ

2.1 Суть проекта

Проект представляет собой разработку iOS-приложения, с помощью которого пользователь может удобно и безопасно вести учёт своих финансов. В нём предусмотрены экраны авторизации и регистрации, чтобы каждый мог хранить данные под личным аккаунтом, а Firestore от Firebase обеспечивает сохранение и синхронизацию внесённых транзакций. Пользователь может добавлять доходы и расходы, указывать сумму, категорию, дату, описание, а также редактировать или удалять уже внесённые записи. Для более наглядного анализа в приложении есть раздел, где все транзакции отображаются по выбранной категории и сортируются по дате, а благодаря интеграции с Firebase все изменения сразу вносятся в базу данных. Специально настроенная валидация при входе и регистрации делает процесс авторизации более безопасным и понятным, а архитектура на базе MVVM упрощает сопровождение кода и масштабирование приложения в будущем.

2.2 Описание работы над проектом

Наш проект представляет собой полноценное мобильное приложение. Нашим руководителем, Сидаматовым Жасуром, был предложен основной концепт и функции: работа с транзакциями, динамическая диаграмма и подробное отображение данных. Членами команды также были предложены некоторые функции, касающиеся контента и дизайна приложения. На первой встрече были озвучены задачи и разработан примерный план действий, включающий в себя подготовку, создание приложения и защиту. В течение подготовительного этапа каждый из членов команды изучал нужный стек технологий.

Стек инструментов и теории для дизайнеров:

- Figma

- Базовое понимание сочетания цветов
- Ознакомление с дизайнами аналогичных приложений

Стек инструментов и теории для Программистов:

- Swift
- SwiftUI
- Xcode
- Паттерны программирования
- Firebase
- Интеграция Firebase

После изучения необходимой базы, руководителем были равномерно распределены задачи в канбан доске. Первыми к работе приступили дизайнеры. Ими была подобрана цветовая гамма, разработан и предложен дизайн экранов приложения и вспомогательных элементов (см. рис. 1, 2, 3). Для экономии времени, дизайны экранов передавались команде программистов по мере готовности. Всего предстояло реализовать 6 экранов: экран авторизации, экран регистрации, экран с доходами, экран с расходами, экран добавления транзакций и экран для просмотра операций по категории. Помимо создания пользовательского интерфейса для каждого из экранов, важной частью проекта являлась настройка Firebase и интеграция бэкенд-решений в само приложение, чтобы данные пользователей сохранялись и синхронизировались в режиме реального времени.

Для отслеживания прогресса выполнения задач активно использовались канбан доска и GitHub репозиторий, куда загружались готовые решения. Для каждой загрузки был написан отчет, отражающий саму проблему, технологии, примененные для ее решения и описание процесса работы. Руководителем проекта был вручную проведен код-ревью для всего загруженного кода. Он оставлял свои замечания и правки, исправление которых улучшало приложение или повышало читаемость кода.

2.3 Мои задачи

Каждому члену команды программистов был присвоен свой набор задач. Мной были выполнены следующие: настройка Firebase, подключение бэкенд-решений к приложению, верстка экрана просмотра транзакций по категории и настройка валидации данных при процессах авторизации. Выполнение этих задач являлось важным этапом разработки приложения, так как от корректной интеграции Firebase зависела вся логика хранения и обмена данными, а реализация экрана просмотра транзакций напрямую влияла на удобство анализа расходов и доходов пользователями.

В первую очередь требовалось настроить проект в консоли Firebase, чтобы приложение могло корректно взаимодействовать с базой данных в режиме реального времени. Я интегрировал необходимые библиотеки с помощью Swift Package Manager а также настроил конфигурационные файлы в Xcode, чтобы в дальнейшем упростить чтение и запись данных.

- Консоль Firebase: создал проект, добавил iOS-приложение, сгенерировал GoogleService-Info.plist для интеграции.
- Подключение бэкенда: с помощью Firestore обеспечил структуру для хранения и обработки данных о доходах и расходах.
- Тестовые записи: добавил несколько тестовых документов, чтобы проверить работоспособность чтения и записи данных из приложения.

Помимо шагов, описанных выше, были написаны на языке Swift сервис-классы для интеграции функционала Firebase в приложение. Написанный код позволяет пользователям регистрироваться, заходить в свой аккаунт и выходить из него, а также управлять своими транзакциями: добавлять, удалять, изменять их. Каждая операция для взаимодействия с Firebase оборачивается в обработчик, возвращающий либо успешный результат, либо ошибку. Ошибки логируются в консоль, что упрощает отладку и поиск причин сбоев. Пользователю при этом выводится короткое уведомление о проблеме – алерт-сообщение.

Следующей задачей стояла верстка экрана просмотра транзакций по категории (см. рис. 4). На главном экране пользователь видит список категорий расходов и общую сумму по каждой из них. При нажатии на строку с конкретной категорией приложение переходит к экрану, где при помощи SwiftUI List отображаются все транзакции выбранной категории. В каждой строке пользователь видит: наименование, описание, сумму и дату транзакции. Кроме того, реализована возможность удаления записей через жест смахивания, благодаря функции onDelete. При этом транзакция удаляется не только из интерфейса, но и из Firestore с помощью вызова `dataService.deleteTransaction`. Для удобства пользователя транзакции автоматически сортируются по дате благодаря методу `sortedTransactions`, использующему `DateFormat`. При нажатии на любую транзакцию доступно её редактирование: выбирается текущий элемент (`selectedTransaction`), и приложение переходит к `AddTransactionView`, где уже заполнены соответствующие поля.

Обновление данных на экране происходит динамически: в коде используется метод `onChange`, который отслеживает любые изменения в категориях и синхронизирует локальный список транзакций. Доступ к сервис-классам для работы с Firebase организован по принципу MVVM, что позволяет:

- Получать и отображать актуальные данные в режиме реального времени,
- Сохранять логику удаления, изменения и добавления транзакций в отдельном слое (ViewModel),
- Отслеживать и логировать возможные ошибки в консоль для упрощения отладки.

Такой подход обеспечивает целостность данных и даёт пользователю удобную возможность анализировать все операции в разрезе конкретных категорий.

Следующей задачей стала настройка валидации данных при процессах авторизации. Она была настроена на двух экранах:

1. AuthorizationView – экран входа в учётную запись. Пользователю предлагается ввести электронную почту (email) и пароль. После успешной авторизации загружаются пользовательские данные (транзакции, категории и т.д.), и выполняется переход на главный экран приложения.

2. RegistrationView – экран создания нового аккаунта. Пользователь вводит email, пароль и подтверждение пароля. В случае успеха сразу происходит вход с вновь созданными учётными данными и переход к основному функционалу приложения.

Оба экрана объединяет единая концепция: пользователь вводит данные, которые проверяются как на стороне приложения, в части логических проверок, так и на стороне Firebase: формат email, уникальность адреса, длина пароля и т.п. Если что-то идёт не так, отображаются понятные сообщения об ошибках, а успешный сценарий открывает пользователю основные разделы приложения.

В AuthorizationView после ввода данных можно нажать кнопку “Войти”. Основная проверка осуществляется сервисом Firebase: если email имеет неверный формат или пароль некорректен, Firebase возвращает соответствующую ошибку. Для известных ошибок используется словарь loginErrorMessages, в котором хранится сопоставление системного текста с локализованным описанием на русском языке. Если ошибка не найдена в словаре, пользователю выводится её исходный текст.

На экране RegistrationView пользователь вводит: E-mail, пароль и подтверждение пароля. Как и в AuthorizationView, здесь есть контроль ошибок, индикатор загрузки и флаг навигации, для перехода к главному экрану.

В обоих экранах реализован единый подход к выводам сообщений об ошибках:

– Алерты (Alert) информируют пользователя о сбое (неверный формат email, неправильный пароль, email уже используется и т.д.).

- Логи в консоли помогают разработчикам и тестировщикам быстро находить причины ошибок на этапе отладки:

- Улучшенный UX: пользователь видит как “дружественные” сообщения (на русском языке), так и индикатор загрузки при более долгих операциях с сетью.

Таким образом, при неверных данных процесс прекращается с ясным уведомлением, а все детали сбоя можно увидеть в отладочной консоли.

Хотя приведённый код сосредоточен в SwiftUI-вью, основная логика работы с Firebase (регистрация, логин, обработка ошибок) вынесена в сервис-класс FirebaseAuthService. Это соответствует концепции MVVM:

- Model – объекты и данные, получаемые из Firebase (например, информация о транзакциях, категориях, пользователе).

- ViewModel – прослойка, где описывается бизнес-логика, обработка ошибок, вызовы сервисов Firebase. В данном случае можно рассматривать appViewModel, DataService и FirebaseAuthService как часть ViewModel и модельной логики.

- View – SwiftUI-экраны (AuthorizationView, RegistrationView) отображают данные и реагируют на действия пользователя, вызывая методы из сервис-классов.

Это упрощает сопровождение кода и делает приложение более масштабируемым: при необходимости, например, заменить Firebase на другой бекэнд, основные изменения произойдут только в модели и сервисах, а экраны останутся практически неизменными.

Благодаря такому разделению, а также системе осмысленных сообщений об ошибках, пользователи получают понятный и удобный процесс авторизации и регистрации.

- Простая валидация: совпадение паролей, минимальная длина пароля, формат email, защищает от заведомо некорректных данных.

- Firebase отвечает за более тонкие проверки: наличие email в базе, валидный ли пароль и возвращает понятные коды ошибок.

– Алерты и логирование позволяют своевременно реагировать на сбои, а при отладке – находить проблемные места в коде.

2.4 Анализ работы

В процессе работы над проектом моя основная деятельность была сосредоточена на интеграции Firebase и обеспечении корректной работы приложения с данными, а также на создании отдельных экранов интерфейса (в частности, экрана просмотра транзакций по категориям).

2.4.1 Что получилось

– Интеграция Firebase: удалось корректно подключить Firebase в проект через консоль Firebase и Swift Package Manager, настроить Firestore и FirebaseAuth, а также обеспечить хранение и обработку данных: транзакций, категорий и пользователей.

– Реализация экрана транзакций по категории: создал экран с использованием SwiftUI List, где пользователь видит все свои расходы или доходы по выбранной категории, может удалять и редактировать транзакции с синхронизацией данных в режиме реального времени.

– Валидация данных: в экранах авторизации и регистрации предусмотрена проверка корректности email и пароля. Также обрабатываются и логируются ошибки, возвращаемые Firebase, с пояснениями для пользователя.

2.4.2 Основные трудности

– Работа с Firebase: наиболее сложным моментом стало разобраться, как правильно организовать структуру данных в Firestore. Приходилось экспериментировать и вносить корректировки, так как структура делалась с нуля.

– Обработка ошибок: Firebase возвращает разные форматы сообщений об ошибках, и не все из них документированы явно. Пришлось искать в

исходном коде библиотек и официальной документации, чтобы корректно переводить эти сообщения для пользователя.

- Согласование дизайна и функционала: иногда возникали задержки, пока дизайнеры не предоставляли обновлённые макеты с внесенными корректировками.

2.4.3 Что не получилось

- Настройка прав доступа через консоль Firebase, на изучение этого не хватило времени.

- Глубокая проверка пароля: например, на использование специальных символов или комбинирование цифр и букв.

В целом удалось придерживаться канбан-доски и распределять задачи между участниками команды. Чуть большее, времени, чем ожидалось ушло на рассмотрение, обсуждение и внесение корректировок, как в код, так и в дизайн

2.4.4 Чему я научился за время проекта

- Базово освоил Swift и SwiftUI.

- Разобрался в архитектуре MVVM: отделение логики от вью, использование сервис-классов для работы с Firebase, внедрение моделей.

- Работа с Firebase: научился создавать, читать, обновлять и удалять записи в Firestore, подключать Firebase Auth и корректно обрабатывать возможные ошибки.

- Получил опыт настоящей командной работы: стал лучше понимать, как работать с Git и канбан-инструментами, чтобы не допускать конфликтов при объединении кода и успевать в сроки.

2.5 Взаимодействие с командой

Взаимодействие с командой проходило в формате регулярных созвонов, которые назначал наш руководитель. Как правило, мы созванивались от одного до трёх раз в неделю, в зависимости от текущей загруженности и количества открытых вопросов. В ходе встреч обсуждались актуальные задачи, уточнялись детали реализации, а также предлагались новые идеи по улучшению интерфейса или функционала приложения. Общение происходило в мессенджере Telegram.

На созвонах каждый член команды отчитывался о проделанной работе, делился успехами и сложностями. Если кто-то сталкивался с технической проблемой или не мог справиться с задачей, наш руководитель помогал её решить. Такая практика помогала быстрее продвигаться по намеченному плану. Благодаря регулярным встречам нам удавалось быть в курсе процесса выполнения задач, своевременно получать обратную связь от руководителя и, при необходимости, корректировать план работы.

2.6 Взаимодействие с руководителем проекта

Руководитель проекта всегда оперативно отвечал на возникающие вопросы и предоставлял подробные разъяснения при обсуждении задач. При необходимости он дополнительно созванивался для ответов на вопросы. Формат общения был рабочим, но при этом без лишнего официоза. Такой подход позволял быстро устранять неопределённости и поддерживать темп работы, так как в любой момент можно было обратиться к руководителю за помощью.

2.7 Оценка работы руководителя

Руководитель проекта своевременно формировал задачи, планировал встречи и проявлял гибкость в решении возникающих вопросов. Он обеспечивал доступ к необходимым ресурсам, делился с нами ссылками на полезные материалы, руководства и инструменты, оперативно давал обратную

связь и помогал в спорных моментах. Благодаря такому подходу команда могла эффективно распределять обязанности, а возникавшие задержки или трудности разработки быстро устранялись. Всё это способствовало стабильному темпу работы и позволяло сосредоточиться на выполнении конкретных задач без длительных простоев.

3 ЗАКЛЮЧЕНИЕ

В ходе проекта была достигнута основная цель — создание мобильного приложения на iOS для удобного сохранения и анализа личных финансов. Все поставленные задачи по дизайну и разработке выполнены: создана система учёта доходов и расходов, реализованы экраны авторизации и регистрации, а также обеспечено взаимодействие с Firebase для хранения данных. Благодаря этому у приложения есть вся необходимая функциональность, чтобы пользователи могли эффективно планировать свой бюджет.

Вклад, который я внёс в достижение цели, заключался в:

- Настройке Firebase и интеграции его сервисов в приложение: аутентификация, хранение данных, обмен информацией в реальном времени.
- Верстке экрана просмотра транзакций по категории и обеспечении возможности их редактирования и удаления.
- Настройке валидации данных при регистрации и авторизации для повышения удобства и надёжности приложения.

Таким образом, основная идея проекта успешно воплощена, и приложение обладает всем необходимым функционалом для удобного учета и анализа финансовых операций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация Xcode // Apple Developer: сайт. – URL: <https://developer.apple.com/documentation/xcode> (дата обращения: 03.11.24).
2. Introducing SwiftUI | Apple Developer Documentation // Apple Developer: сайт. – URL: <https://developer.apple.com/tutorials/swiftui/> (дата обращения: 04.11.24).
3. Human Interface Guidelines | Apple Developer Documentation // Apple Developer: сайт. – URL: <https://developer.apple.com/design/human-interface-guidelines> (дата обращения: 06.11.24).
4. Hacking With Swift // Paul Hudson: сайт. – URL: <https://www.hackingwithswift.com/> (дата обращения 06.11.24)
5. Документация Firebase // Google Firebase: сайт. – URL: <https://firebase.google.com/docs> (дата обращения 15.11.24)

ПРИЛОЖЕНИЕ

Техническое задание к проекту «Разработка мобильного приложения на iOS для учета трат»

Цель: разработать мобильное приложение, которое позволит пользователям вести учёт личных доходов и расходов, сохранять транзакции в удобной форме и анализировать структуру трат.

Руководитель проекта: Сидаматов Жасур Илхомович

Термины и сокращения:

- iOS - операционная система для мобильных устройств компании Apple.
 - SwiftUI – это декларативный фреймворк Apple для создания пользовательских интерфейсов на всех платформах Apple с использованием минимального кода и обновления UI в реальном времени.
 - Core Data – это фреймворк Apple для управления моделью данных в приложениях, обеспечивающий сохранение, выборку и управление объектами в удобной объектно-ориентированной форме.
 - GitHub/GitLab - это платформы для управления репозиториями исходного кода с использованием Git, предоставляющие инструменты для совместной разработки.
 - МП - мобильное приложение
 - Техническое задание (ТЗ) — это документ, в котором описываются требования и спецификации для выполнения проекта или разработки продукта.
 - Pull request (PR) — это предложение изменений в репозитории, которое отправляется на рассмотрение для слияния с основной веткой проекта после ревью и одобрения.
 - Firebase - это платформа от Google, предоставляющая набор инструментов и сервисов для разработки мобильных и веб-приложений, включая базы данных, аутентификацию, хостинг, аналитику и уведомления в реальном времени.
- Верстка экрана — это процесс разработки и расположения элементов интерфейса (текстов, кнопок, изображений и других компонентов) на экране

приложения или веб-страницы с учетом дизайна, функционала и удобства пользователя.

– Flow пользователя в приложении (или пользовательский поток) — это последовательность шагов и действий, которые пользователь выполняет внутри приложения для достижения своей цели, например, регистрации, совершения покупки или выполнения задачи. Это включает в себя взаимодействие с интерфейсом, переходы между экранами и обработки действий, направленных на выполнение конкретной функции.

Технические требования

Таблица 1 - функциональные и нефункциональные требования

Техническое требование	Язык разработки	СУБД	Потребители
Минимальная версия поддержки iOS 15	Swift		iOS разработчики
Технологии для разработки МП	Xcode, Swift, SwiftUI, URLSession, SPM.	CoreData, Firebase	iOS разработчики
Система контроля версий GitHub/GitLab	GitHub/GitLab		iOS разработчики
Дизайн приложения должен соответствовать Apple HIG, обеспечивая интуитивный интерфейс, минимализм,	Figma		Дизайнеры

адаптацию под разные экраны мобильного устройства iOS			
Возможность регистрации/авторизации	Firebase		iOS разработчики
Реализация flow пользователя в макетах Figma	Figma		
Поддержка оффлайн режима МП. В оффлайн режиме пользователь остается авторизованным на своей учетной записи и может добавлять новые транзакции, они сохраняются в локальную базу данных. Как только появляется интернет система должна обновить данные в Firebase.	Swift		iOS разработчики
Выбрать архитектуру для МП с учетом дальнейшего расширения функциональности, учитывая что используется фреймворк SwiftUI.	Swift		iOS разработчики

Содержание работы

Этапы задач:

- 1) разработка технического задания. 1-10 ноября. Результат - Текстовый форма,
- 2) проектирование дизайна приложения. 10-30 ноября. Результат - Проект в Figma,
- 3) разработка МП. 20 ноября - 10 декабря. Результат - Проект на GitHub,
- 4) формирование отчета и презентации к защите. 10 декабря - 15 декабря. Результат - Отчет и презентация.

Задачи

Таблица 2 - Задачи

Название задачи	Описание	Technical Specification	Этап
Создать техническое задание	<p>Создать техническое задание в соответствии с требованиями представленные на лекция по предмету "Управление проектами по разработке мобильных и сетевых приложений".</p> <p>Требования и шаблон - https://docs.google.com/document/d/1P38OYJh_oECpPF4sMvkvrdekUI7h06dvl/edit.</p> <p>Итог: готовое ТЗ заполненное на странице проекта.</p>	Разработка мобильного приложения на iOS для учета затрат	Разработка технического задания. 1-10 ноября. Результат - Текстовый формат.

Создать начальн ый файл с дизайно м и дать доступ к нему	Создать исходный файл с дизайном и предоставить его команде разработки. Итог: ссылка на проект в Figma в чате группы	Разраб отка мобил ьного прило жения на iOS для учета трат	Проекти рование дизайна приложе ния. 10- 30 ноября. Результа т - Проект в Figma
Подобра ть цветову ю гамму МП	По результатам анализа конкурентов и изучению цветовых гамм, определить основные цвета для МП. Итог: цветовая гамма будущего МП	Разраб отка мобил ьного прило жения на iOS для учета трат	Разработ ка техничес кого задания. 1-10 ноября. Результа т - Текстов ый формат.

Спроектировать дизайн экрана "Авторизация"	<p>Создать визуальный дизайн экрана для входа пользователя в систему. Экран должен включать поля для ввода логина и пароля, кнопки входа и регистрации нового пользователя. Дизайн должен быть интуитивно понятным, обеспечивать легкость восприятия и соответствовать общей стилистике приложения.</p> <p>Итог: дизайн экрана в Figma</p>	Разработка мобильного приложения на iOS для учета трат	Проектирование дизайна приложения. 10-30 ноября. Результат - Проект в Figma
Спроектировать экран "Главная"	<p>На главном экране необходимо отобразить</p> <ul style="list-style-type: none"> • Траты в списочном виде • Кнопка добавления новой транзакции • Кнопка открытия профиля • Диаграмма трат <p>Итог: дизайн экрана в Figma</p>	Разработка мобильного приложения на iOS для учета трат	Проектирование дизайна приложения. 10-30 ноября. Результат - Проект в Figma

<p>Спроектировать дизайн экрана "Добавление транзакции"</p>	<p>Создать удобный и понятный интерфейс для добавления новой финансовой транзакции. Экран должен включать следующие элементы:</p> <ol style="list-style-type: none"> 1. Поля для ввода суммы транзакции и выбора категории (например, доход или расход). 2. Возможность указать дату и время транзакции. 3. Поле для ввода вида траты 4. Поле для добавления заметки или комментария. 5. Кнопку "Сохранить" для подтверждения добавления транзакции. <p>Дизайн должен быть минималистичным, с легким доступом ко всем функциям, и обеспечивать быстрое добавление информации</p>	<p>Разработка мобильного приложения на iOS для учета трат</p>	<p>Проектирование дизайна приложения. 10-30 ноября. Результат - Проект в Figma</p>
<p>Спроектировать дизайн экрана "Профиль"</p>	<p>Экран должен отображать основную информацию о пользователе и возможность выйти из аккаунта.</p> <p>Итог: дизайн экрана в Figma</p>	<p>Разработка мобильного приложения на iOS для учета трат</p>	<p>Проектирование дизайна приложения. 10-30 ноября. Результат - Проект в Figma</p>

Подключить к проекту возможность работы с Firebase	<p>Добавить зависимость через SPM в проект.</p> <p>Добавить возможность авторизации и хранения данных в Firebase - создать отдельные сервисы, которые в дальнейшем будут использованы системой.</p> <p>Итог: Итог: PR с описанием выполненной задачи</p>	Разработка мобильного приложения на iOS для учета трат	Разработка МП. 20 ноября - 10 декабря. Результат - Проект на GitHub
Спроектировать архитектуру МБ	Подобрать архитектуру приложения под фреймворк SwiftUI.	Разработка мобильного приложения на iOS для учета трат	Разработка МП. 20 ноября - 10 декабря. Результат - Проект на GitHub

Сверстать экран авторизации	<p>Выполнить верстку экрана "Авторизации" и "Регистрации" по спроектированному макету Figma.</p> <p>Макет: https://www.figma.com/design/wSfx7SNCRwGH70y4Oh9OdG/MoneyManager?node-id=0-1&node-type=canvas&t=J1zlDQ8N4hlThmp9-0</p> <p>Итог: Итог: PR с описанием выполненной задачи</p>	Разработка мобильного приложения на iOS для учета трат	Разработка МП. 20 ноября - 10 декабря. Результат - Проект на GitHub
Сверстать экран "Главная"	<p>Сверстать экран "Главная" по макету Figma.</p> <p>Макет - https://www.figma.com/design/wSfx7SNCRwGH70y4Oh9OdG/MoneyManager?node-id=0-1&node-type=canvas&t=J1zlDQ8N4hlThmp9-0</p> <p>Итог: PR с описанием выполненной задачи</p>	Разработка мобильного приложения на iOS для учета трат	Разработка МП. 20 ноября - 10 декабря. Результат - Проект на GitHub

Сверстать экран "Добавление транзакции"	<p>Сверстать экран "Добавление транзакции" по Figma.</p> <p>Макет - https://www.figma.com/design/wSfx7SNCRwGH70y4Oh9OdG/MoneyManager?node-id=0-1&node-type=canvas&t=J1zlDQ8N4hlThmp9-0</p> <p>Итог: PR с описанием</p>	Разработка мобильного приложения на iOS для учета трат	<p>Разработка МП.</p> <p>20 ноября - 10 декабря.</p> <p>Результат - Проект на GitHub</p>
Сформировать отчет и презентацию для защиты проекта	<p>Сформировать отчет и презентацию для защиты проекта.</p> <p>Отчет должен соответствовать ГОСТ 7-32. 2017.</p> <p>Итог: отчет и презентация</p>	Разработка мобильного приложения на iOS для учета трат	<p>Формирование отчета и презентации к защите.</p> <p>10 декабря - 15 декабря.</p> <p>Ответственный - Атрашкевич Д.Д.</p> <p>Результат - Отчет и презентация</p>

Основные результаты работы

Мобильное приложение для учета доходов и расходов с функционалом анализа трат.

В результате разработки был создан прототип приложения, который включает:

- мобильное приложение: полностью функциональное приложение для учета доходов и расходов с возможностью анализа трат,
- проект на GitHub: исходный код приложения, доступный для просмотра и дальнейшего развития,
- отчет по разработке: подробный документ, описывающий этапы разработки приложения, использованные технологии и методы,
- презентация: подготовлена презентация для защиты проекта, включающая описание функционала и результатов работы.

Рисунок 1 (Главный экран)

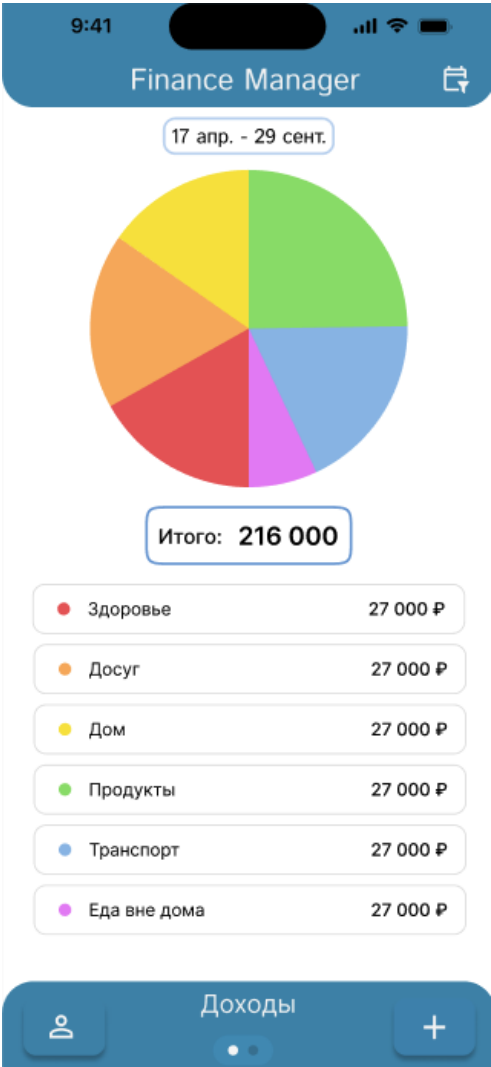


Рисунок 2 (Экран добавления транзакции)

9:41

📶 🔋

Назад

Finance Manager

Название траты

Сумма


₽


Дата


Jul 21, 2023


Расходы


Доходы


 Здоровье


 Досуг


 Дом

 Продукты

 Еда вне дома

 Транспорт

 Образование

 Подарки

...

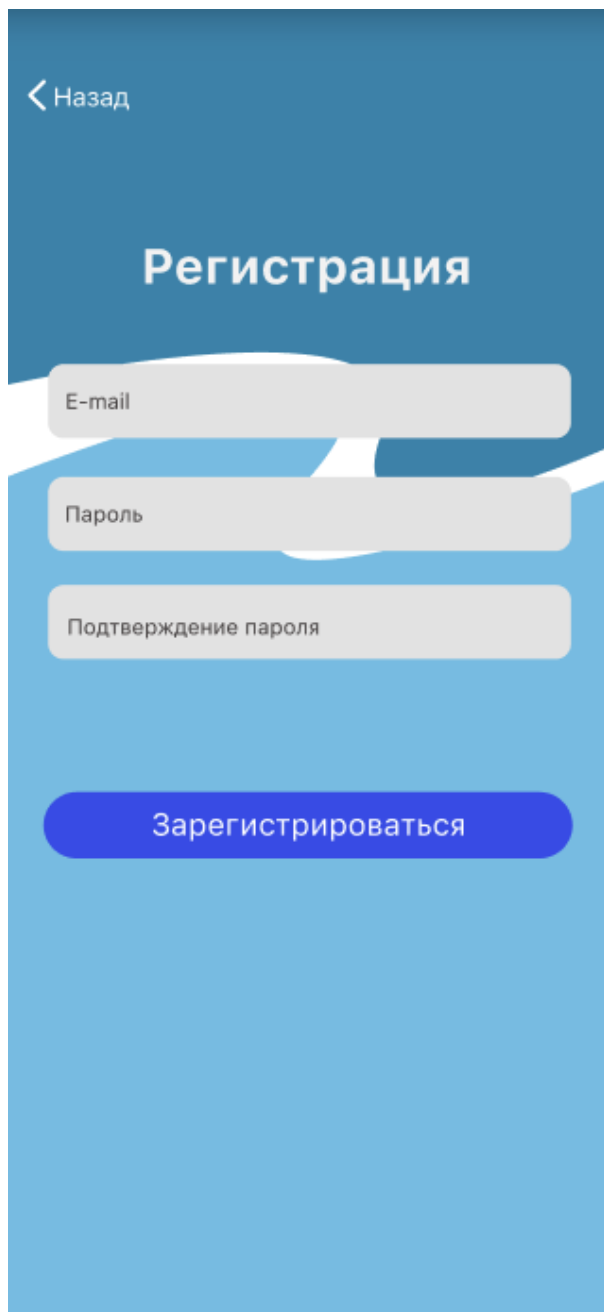
Еще

Описание

Напишите здесь свой комментарий к трате

Готово

Рисунок 3 (Экран регистрации)



The image shows a mobile application registration screen. It features a dark blue header with a back arrow and the text 'Назад'. Below the header, the word 'Регистрация' is displayed in large white font. The form consists of three light gray input fields with rounded corners, labeled 'E-mail', 'Пароль', and 'Подтверждение пароля'. At the bottom of the form is a prominent blue button with rounded corners labeled 'Зарегистрироваться' in white text. The background of the screen is a light blue gradient with a subtle white abstract shape.

< Назад

Регистрация

E-mail

Пароль

Подтверждение пароля

Зарегистрироваться

Рисунок 4 (Экран категории)

