МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра дифференциальных уравнений и математической экономики

Разработка iOS-приложения «Личный финансовый помощник»

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 441 группы

направления 09.03.03 Прикладная информатика

механико-математического факультета

Пантелеева Дмитрия Александровича

Научный руководитель
зав. кафедрой, д.ф.-м.н.,
профессор _____ С.И. Дудов
подпись, дата

Зав. кафедрой
зав. кафедрой, д.ф.-м.н.,
профессор _____ С.И. Дудов
подпись, дата

Саратов 2023

Введение. В настоящее время все больше людей сталкиваются с проблемой управления своими личными финансами. В связи с этим появляется необходимость в инструментах, с помощью которых можно просто и удобно отслеживать свои доходы и расходы, сберегать деньги и получать рекомендации по управлению семейным бюджетом. В данном контексте, разработка iOS- приложения «Личный финансовый помощник» является актуальной и востребованной темой.

Основной целью приложения является помощь пользователям в учете своих доходов и расходов, а также в формировании сберегательной культуры и эффективного управления своим денежным бюджетом. Все это должно сделать жизнь пользователей более комфортной и удобной.

Важным элементом "Личного финансового помощника"является аналитика, которая поможет пользователям анализировать свои расходы, выявлять места излишних трат и планировать свой семейный бюджет более эффективно.

Таким образом, введение данной технологии имеет высокую актуальность, привнося возможность сбережения денежных ресурсов и повышения финансовой грамотности пользователей.

Первый раздел посвящен обзору аналогичных программных продуктов.

На сегодняшний день на рынке мобильных приложений существует множество приложений по учету личных финансов. Рассмотрим несколько наиболее популярных приложений и выявим их недостатки, которые не удовлетворяют потребности пользователей.

1. Приложение "Счетчик расходов Money Pro"

   Money Pro – это iOS-приложение для учета личных финансов, которое объединяет в себе возможности управления финансами и бюджетом. Основные функции приложения: учет доходов и расходов, возможность добавления кредитов, инвестиций и активов. Однако, основным недостатком приложения "Money Pro"является высокая цена для полного функционала и крайне запутанный интерфейс. Приложение ориентировано на продвинутых пользователей, которые готовы тратить деньги на использование всей функциональности. Интерфейс Money Pro мо-

жет вызывать трудности для новых пользователей, так как там совсем не понятная навигация и слишком много элементов меню.

2. Приложение "Monefy - Личный финансовый менеджер"
   Monefy – это приложение для учета личных финансов, которое имеет простой и понятный интерфейс. Он позволяет просто и быстро записывать все расходы и доходы. Существенным недостатком приложения Monefy является отсутствие возможности синхронизировать данные с облачными сервисами. По данной причине пользователь, ведущий учет на одном устройстве, никак не сможет переместить свои данные на другое устройство.
3. Приложение "Wallet"
   Wallet – это приложение для учета личных финансов, которое предоставляет возможность создавать и управлять счетами, записывать транзакции и формировать отчетность. Однако, у данного приложения так же есть ряд недостатков. Основными из них являются сложная настройка параметров и трудность в использовании. Ориентировано на опытных пользователей, которые могут управлять несколькими банковскими счетами и требуют множества дополнительных функций.

Второй раздел посвящен выбору инструментов для разработки приложения. В данном разделе описан и аргументирован выбор инструментов для разработки мобильного приложения, серверной части мобильного приложения и библиотек, фреймворков.

Для проектирования мобильного приложения был выбран язык Swift. Преимущества данного языка программирования:

- Интеграция с платформой Apple. Swift был разработан Apple и является основным языком для создания iOS-приложений. Swift имеет широкую поддержку и активное сообщество разработчиков, что обеспечивает более быструю разработку приложений для iOS;
- Безопасность. Swift разработан с учетом данного критерия с целью предотвращения ошибок безопасности и сокращения рисков, связанных с ними;
- Быстрое развертывание. Swift обеспечивает быстрое развертывание приложений на iOS-устройствах благодаря наличию компилятора, ко-

торый может создавать оптимизированный машинный код непосредственно для устройства;
- Разработка в Xcode. Swift поддерживается в Xcode - интегрированной среде разработки, которая используется для создания iOS-приложений.

Таким образом, Swift является лучшим инструментом для реализации клиентской части приложения.

Для проектирования серверной части приложения был выбран язык PHP. Данный язык выбран по следующим причинам:

- Широкая популярность. PHP является одним из самых распространенных языков программирования для разработки серверных приложений. Это обеспечивает большой выбор инструментов и активное сообщество разработчиков, что позволяет нам быстро и эффективно разрабатывать серверную часть нашего приложения;
- Удобство использования. PHP имеет простой и понятный синтаксис, что делает его более доступным для новых разработчиков. Кроме того, PHP обеспечивает легкость внесения изменений в код, что может быть полезно при изменении функциональности приложения в будущем;
- Хорошая поддержка баз данных. PHP легко интегрируется с большинством баз данных, в том числе MySQL и PostgreSQL;
- Быстрая разработка. PHP разработка может быть быстрой, особенно в сравнении с некоторыми другими языками программирования. Это может оказаться критичным при разработке серверной части приложения, которая должна быть быстрой и эффективной;
- Большое количество готовых решений. PHP имеет широко известные фреймворки, такие как Laravel, Symfony и CodeIgniter, которые предоставляют готовые решения для работы с базами данных, обработки запросов и диспетчеризации маршрутов. Они облегчают разработку и могут сократить время создания серверной части.

Для разработки клиентской части было принято решение использовать следующие библиотеки и фреймворки: SwiftUI, Combine, Stinsen и Swinject.

SwiftUI, Combine и Stinsen нужны для удобной и быстрой разработки пользовательского интерфейса. SwiftUI – библиотека компонентов для удобной разработки графического интерфейса в декларативном стиле. Combine

– это фреймворк от Apple, позволяющий использовать функционально-реактивный стиль программирования, что увеличивает скорость разработки в несколько раз. Stinsen – это библиотека с открытым исходным кодом, реализующая удобную работу с навигацией между экранами. A Swinject – это библиотека, помогающая хранить и управлять внешними зависимостями, с помощью данной библиотеки мы реализуем шаблон проектирования «внедрение зависимостей».

Для разработки серверной части было принято решение использовать фреймворк Symfony. Symfony предлагает быструю разработку и управление приложениями, позволяет легко решать рутинные задачи программиста.

Третий раздел посвящен разработке приложения. Данный раздел состоит из трех составляющих:

1. Дизайн приложения.

   Первым шагом необходимо разработать дизайн-систему, которая позволит гораздо быстрее реализовать клиент.

2. Разработка.

   В данном пункте необходимо разработать основные модули приложения. На этом этапе было решено разрабатывать сначала серверную часть модуля, затем тот же модуль на клиенте.

3. Тестирование.

   Сразу после разработки необходимо протестировать каждый модуль на корректную работу.

Разработка дизайн-макета перед разработкой приложения является важным этапом, поскольку позволяет проектировать и оценивать пользовательский интерфейс и функционал. В свою очередь, это позволяет предотвратить возможные ошибки и улучшить пользовательский опыт. В результате применения данного подхода удается сократить время на разработку и улучшить качество конечного продукта.

По этим причинам в работе в первую очередь был разработан дизайн-макет. На рисунке 1 можно увидеть результат работы над данным макетом.

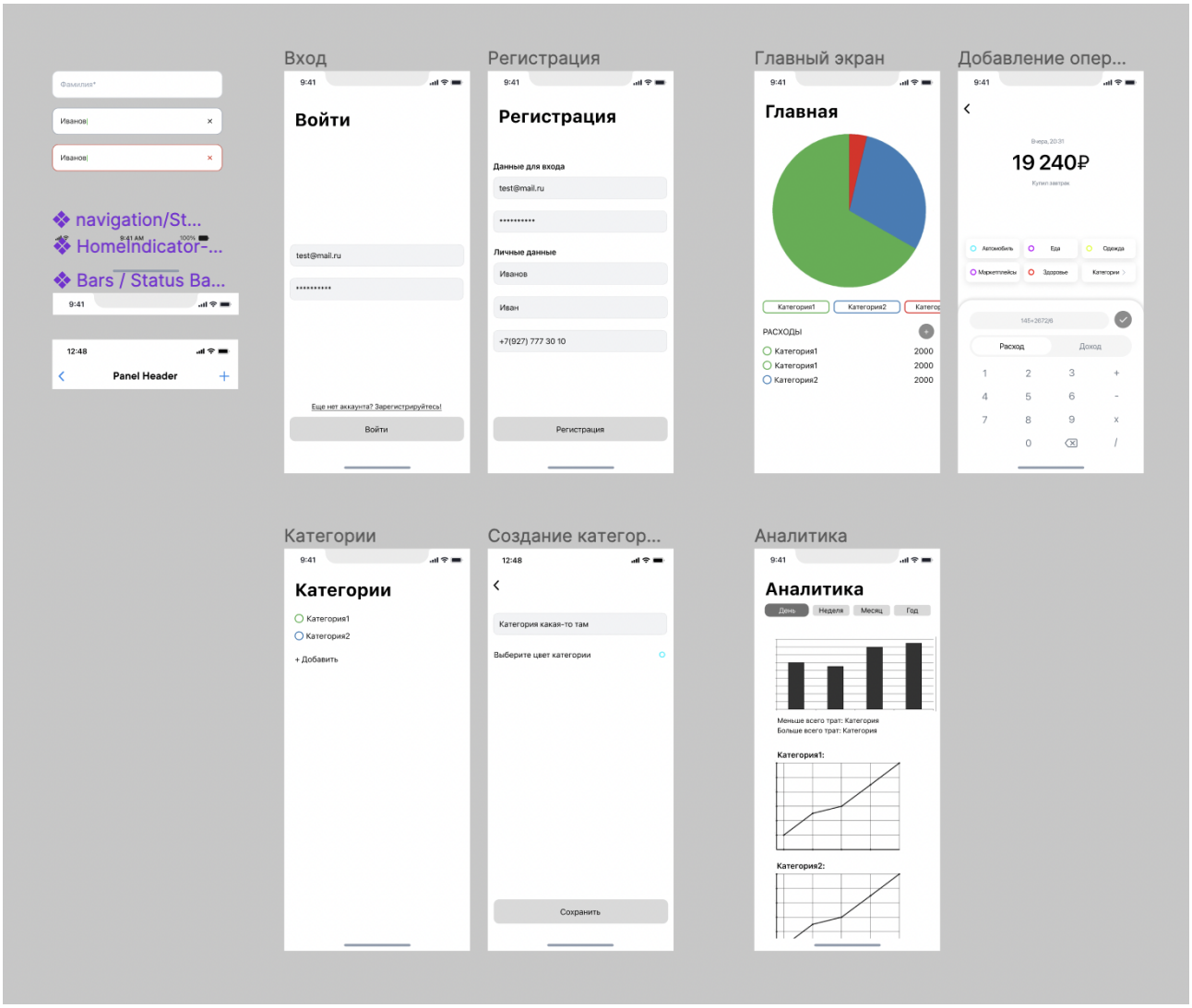

Рисунок 1 — Дизайн-макет

Следующим этапом работы является разработка самого приложения: его серверной части и клиентской.

Для разработки мобильного приложения было принято решение использовать архитектуру MVVM.

MVVM позволяет связывать элементы View со свойствами и событиями ViewModel. При этом ViewModel — абстракция представления. В MVVM есть:

- View — содержит поля, соответствующие интерфейсу пользователя;
- ViewModel — содержит такие же поля, но в предметной области;
- Model - содержит структуры данных, используемые в приложении.

Разработка на данной архитектуре позволяет очень комфортно разбивать программу на модули. Таким образом, в этой работе разработка велась модульно.

Самым важным модулем в работе является модуль авторизации и регистрации. Для этого модуля на серверной части приложения был создан SecurityService, который отвечает за вход или регистрацию пользователей. В мобильном приложении был реализован экран регистрации и экран входа, которые динамически меняются между собой в зависимости от намерений пользователя. Результат показан на рисунке 2.

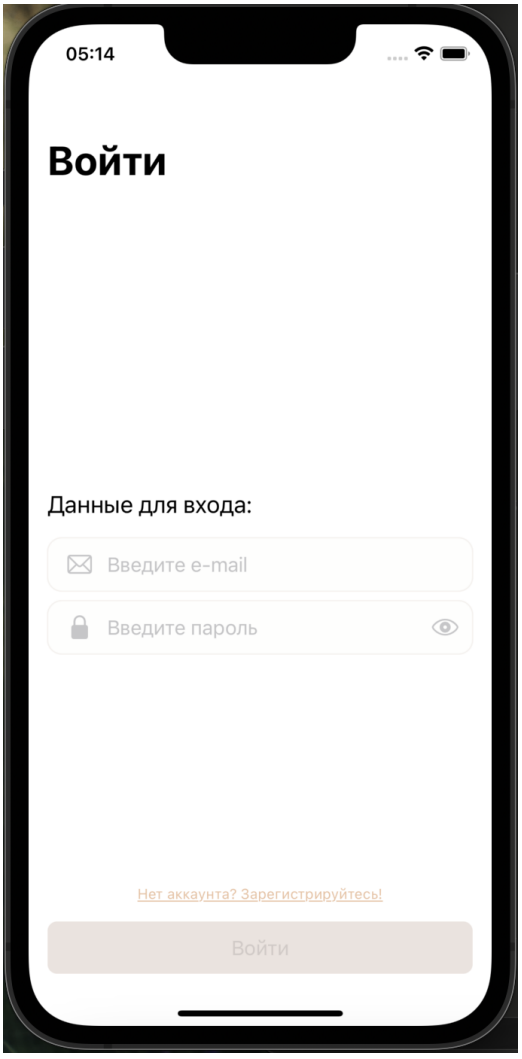

Рисунок 2 — Экран входа в приложение

Вторым по важности модулем является основной модуль. В этом модуле пользователь может ознакомиться с общей информацией о своих тратах и последних транзакциях. В мобильном приложении были реализованы круго-

вая диаграмма, горизонтальный список категорий и список транзакций (рис. 3). На серверной части были реализованы API-методы для получения этих данных.

Рисунок 3 — Главный экран

Для более детального анализа в приложении был реализован модуль аналитики, который предоставляет пользователю несколько удобных графиков и контроллер, позволяющий выбрать временной интервал, по которому отображаются данные на графиках.

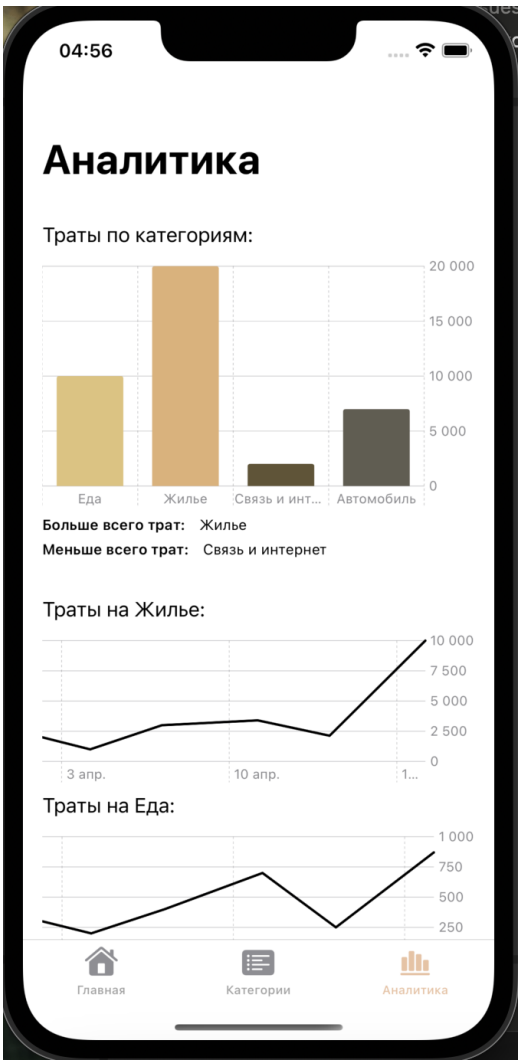

Рисунок 4 — Экран аналитики

И последним шагом в этапе разработки является реализация модуля категорий. Данный модуль отвечает за создание категорий и их отображение для пользователя (рис. 5).

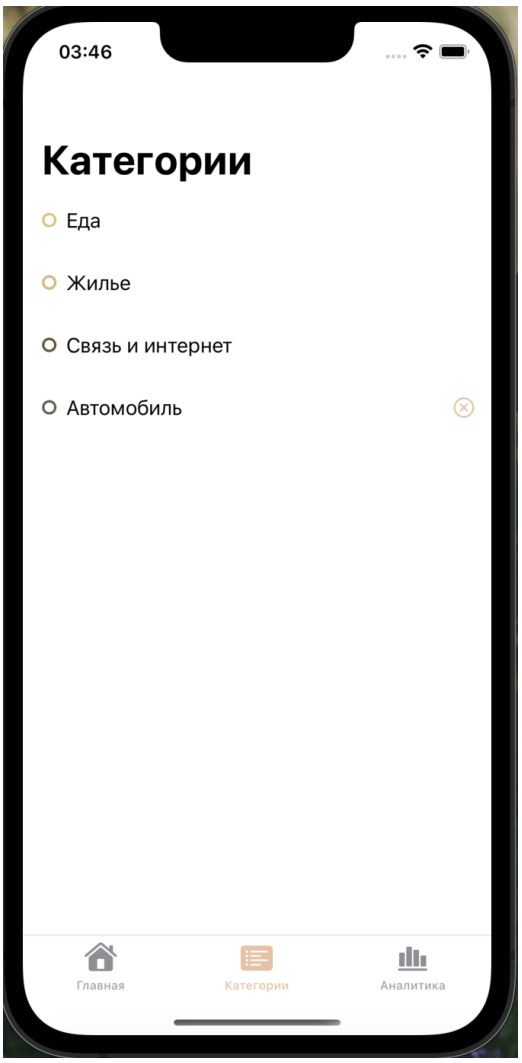

Рисунок 5 — Экран категорий

Заключительным шагом в работе является тестирование. На данном этапе было произведено полное мануальное тестирование функционала серверной части и корректной работы мобильного приложения. Каждый модуль был протестирован отдельно, была подтверждена корректность работы API-методов и правильности отрисовки элементов на экранах мобильного приложения.

Заключение. В ходе выполнения дипломной работы было проведено исследование имеющихся на рынке аналогов, что позволило сформулировать требования к разрабатываемому приложению. Был разработан дизайн-макет приложения.

Основными результатами работы являются создание функционала учета личных финансов и приложения для iOS-платформы. Пользователь имеет

возможность просматривать потраченные им деньги по разным категориям, анализировать свои траты. Вся информация о финансах отображается в удобной пользователю форме, что позволяет ему контролировать свои финансы за определенный период времени.

Оценка полноты решений, поставленных задач, свидетельствует о том, что приложение покрывает все основные функции в учете личных финансов, которые были поставлены перед ним.

Рекомендуется использование результатов работы всем, кто заинтересован в том, чтобы контролировать свои финансы, вести учет и планировать свой бюджет.

Оценка эффективности предложенных решений показала, что приложение имеет несколько преимуществ перед аналогами. В частности, удобный и простой дизайн, быструю скорость работы, хорошую защиту и широкий функционал.

Сопоставление с лучшими достижениями в данной области позволяет сделать вывод о том, что разработанное приложение является достойным конкурентом, который может удовлетворить потребности большинства пользователей, использующих аналогичные приложения.

Таким образом, выполненная работа имеет все качества MVP-продукта и может конкурировать с другими аналогами на рынке.