МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»
Кафедра математической кибернетики и компьютерных наук

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА SWIFT С БЕКЕНДОМ НА PYTHON ДЛЯ ПОИСКА И БРОНИРОВАНИЯ ЖИЛЬЯ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Парамина Данилы Андреевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2023

Введение.

В современном мире технологии играют все более важную роль во многих аспектах нашей жизни. Мобильные приложения стали ключевым инструментом для множества повседневных операций, включая поиск и бронирование жилья. В связи с этим, актуальность разработки новых и усовершенствование существующих приложений в данной сфере растет с каждым днем.

В этом контексте, дипломная работа сфокусирована на проектировании и реализации мобильного приложения на Swift (SwiftUI) с бекендом на Python (Django) для поиска и бронирования жилья.

Основной целью исследования является создание функционального и удобного в использовании приложения, которое будет отвечать требованиям и ожиданиям пользователей. Достижение этой цели предполагает решение следующих задач:

1. Провести обзор и анализ основных принципов и технологий, используемых в разработке мобильных приложений на Swift и SwiftUI.
2. Изучить возможности и особенности разработки бекенда на Python и Django.
3. Определить требования к новому приложению и разработать его архитектуру.
4. Реализовать и протестировать мобильное приложение и соответствующий бекенд.

В ходе работы предполагается использовать разнообразные методы исследования, включая изучение и анализ научной и технической литературы, изучение открытых источников и ресурсов, разработку программного обеспечения, тестирование и отладку приложения, а также анализ и интерпретацию полученных результатов.

Результаты данного исследования могут быть полезны как для разработчиков и специалистов в области проектирования и разработки программного обеспечения, так и для широкого круга людей, заинтересованных в использовании современных технологий для упрощения процесса поиска и бронирования жилья.

В ходе практической части работы будет проведено проектирование и разработка мобильного приложения с использованием Swift и SwiftUI, а также создание бекенда на Python и Django. Процесс разработки будет включать в себя

этапы планирования, проектирования, кодирования, тестирования и отладки.

Особое внимание будет уделено обеспечению высокого уровня практичности, эффективности и безопасности приложения. Это позволит удовлетворить ожидания пользователей и предоставить им качественный и надежный инструмент для поиска и бронирования жилья.

В заключительной части работы будут сформулированы основные выводы, проведен анализ эффективности разработанного приложения и определены возможные направления для дальнейшего развития проекта.

Таким образом, результаты данного исследования способствуют расширению представлений о современных технологиях и подходах к разработке мобильных приложений, а также о вкладе этих технологий в упрощение и улучшение процесса поиска и бронирования жилья.

Основное содержание работы.

В первой главе рассматриваются технологии и инструменты, которые использовались в процессе разработки приложения.

1. Swift - мультипарадигменный язык программирования, объединяющий концепции из различных стилей программирования. Он гибок и может использоваться для решения разнообразных задач. Синтаксис Swift является легким для чтения и понимания. Безопасность типов и безопасность памяти являются важными чертами языка. Swift поддерживает плейграунды, интерактивную среду для тестирования кода в реальном времени. Он также предоставляет богатую стандартную библиотеку с множеством инструментов. Swift является мощным, безопасным и удобным инструментом для разработки мобильных приложений на платформах Apple.
2. SwiftUI - инновационный фреймворк для разработки пользовательских интерфейсов на платформах Apple. Он был представлен компанией Apple в 2019 году и предназначен для работы с языком программирования Swift. SwiftUI предлагает единый декларативный подход к созданию интерфейсов для iOS, macOS, watchOS и tvOS. Он отличается от предыдущих фреймворков, таких как UIKit и AppKit, тем, что разработчики описывают ожидаемый результат, а не детальные инструкции по созданию и обновлению интерфейса. Весь исходный код и информация о логике, макетах и интерфейсных привязках сосредоточены в одном месте. SwiftUI представляет собой более простой и универсальный подход к разработке интерфейсов на платформах Apple.
3. Python - один из самых популярных языков программирования, благодаря своим множеству преимуществ. Синтаксис Python прост и читаем, что удобно для новичков. Он также является высокоуровневым языком, автоматизирующим низкоуровневые задачи. Python поддерживает объектно-ориентированное и мультипарадигменное программирование. В нем доступна богатая стандартная библиотека, что уменьшает зависимость от сторонних решений. Python портативен и работает на разных платформах. Он интерпретируемый и имеет большое и активное сообщество разработчиков. Python применим во многих областях, таких как веб-разработка, научные исследования, машинное обучение и обработка данных. Его динамическая типизация делает код более компактным, а богатые возмож-

ности для интеграции делают его универсальным инструментом. Python также поддерживает тестирование и обладает простотой и гибкостью, что делает его предпочтительным языком для разработчиков всех уровней.

4. Бекенд-разработка является фундаментом создания мобильных приложений, включая работу с серверной стороной. Она включает создание и поддержку серверного кода, работу с базами данных, реализацию бизнес-логики и управление безопасностью. Без качественного бекенда приложение может столкнуться с проблемами производительности, масштабируемости и безопасности, что негативно отразится на пользовательском опыте и популярности приложения. Бекенд-разработчики обеспечивают стабильность, эффективность и масштабируемость приложения, обрабатывая запросы и защищая пользовательские данные. Они являются спинным мозгом приложения, обеспечивая его успешную работу.
5. Бекенд-разработка включает в себя несколько ключевых компонентов, которые сотрудничают, чтобы обеспечить надежное и эффективное мобильное приложение. Эти компоненты включают серверы, базы данных, API, среды выполнения и фреймворки/библиотеки. Они взаимодействуют друг с другом, обрабатывая запросы, обеспечивая безопасность и улучшая пользовательский опыт. Каждый компонент имеет различные реализации в зависимости от требований, технологий и архитектуры приложения.
6. Используя Django для бекенд-разработки мобильных приложений, вы получаете мощный и гибкий фреймворк с рядом преимуществ. Django предлагает "полную батарею в комплекте включающую ORM, инструменты аутентификации, административный интерфейс и многое другое, что упрощает процесс разработки и сокращает необходимость в сторонних библиотеках. Фреймворк придерживается принципов DRY и "конвенции против конфигурации что уменьшает дублирование кода и упрощает настройку. Django также серьезно относится к безопасности и помогает избегать распространенных уязвимостей. Он масштабируем и способен обрабатывать большие объемы трафика. Кроме того, Django написан на Python, что позволяет использовать его преимущества и богатую стандартную библиотеку.
7. Монолитная архитектура - это подход к организации приложения, при котором все компоненты приложения (фронтенд, бекенд, база данных) раз-

рабатываются, развертываются и работают в виде одного цельного блока или модуля. Это означает, что все части приложения сильно связаны друг с другом и изменения в одной части могут повлиять на все остальные. Преимущества монолитной архитектуры включают простоту разработки и развертывания, а также оптимальную производительность, так как вызовы между компонентами происходят напрямую. Однако у такой архитектуры есть и недостатки. Масштабирование отдельных компонентов может быть сложным, изменения в одной части могут вызывать проблемы в других частях системы, а также возможен более высокий риск сбоев, так как сбой в одной части может повлечь неработоспособность всего приложения. Монолитная архитектура подходит для небольших и средних проектов, где требуется быстрая разработка и развертывание, а также для проектов с ограниченными ресурсами или командами разработчиков, предпочитающих работать с единой кодовой базой.

Во второй главе рассматривается проектирование и реализация системы.

1. Функциональные требования - это конкретные функции и возможности, которые должно иметь приложение. Включает в себя такие аспекты, как вход в систему, управление профилем пользователя, обработка данных и т.д.

   Исходя из определения функциональных требований приложение по поиску и бронирования жилья должно обладать следующими возможностями:

   а) Авторизация.
   б) Регистрация.
   в) Просмотр своего профиля с данными.
   г) Возможность поиска/фильтрации по параметрам и просмотра жилищ.
   д) Бронирования жилища.
   е) Удаление брони.
   ж) Просмотр данных о владельце помещения.

2. Нефункциональные требования - это требования, связанные с производительностью, безопасностью, надежностью и другими аспектами системы, которые не относятся к конкретным функциям приложения. Исходя из определения нефункциональных требований приложение по поиску и бронирования жилья должно обладать следующими возможностями:

а) Доступ к приватной информации пользователя осуществляется только посредством JWT-токенов, что обеспечивает высокий уровень безопасности.

б) Для подтверждения профиля и аутентификации пользователя предусмотрена отправка кода на электронную почту.

в) Для оптимального хранения данных в базе данных используется эффективная система обработки дат.

г) Для ускорения загрузки экранов и предотвращения задержек предусмотрена пагинация.

д) В системе реализована эффективная обработка ошибок для предотвращения сбоев в работе приложения.

е) Для отслеживания действий пользователей и устранения возможных проблем в системе используется логгирование.

3. В проекте была выбрана монолитная архитектура для бекенда на Django и архитектуру MVVM для мобильного приложения на SwiftUI. Монолитная архитектура была выбрана для бекенда из-за простоты развертывания и поддержки, а также из-за относительно небольшого размера проекта. Django предоставляет быструю и надежную разработку с глубокой интеграцией компонентов и богатым функционалом. Для мобильного приложения выбрана архитектура MVVM с использованием SwiftUI, обеспечивающая четкое разделение бизнес-логики и пользовательского интерфейса. Это позволяет достичь высокой производительности и удобства разработки, обеспечивая простоту и согласованность на сервере с монолитной архитектурой и эффективное построение пользовательского интерфейса с помощью MVVM и SwiftUI.
4. Интерфейс пользователя должен быть простым и интуитивно понятным, позволяя пользователям легко взаимодействовать с функциями приложения. Для этого приложение разрабатывается в соответствии со следующими функциональными требованиями:

а) Авторизация: Интерфейс будет включать в себя форму авторизации, которая позволит пользователям вводить свои учетные данные для доступа к своему профилю и функциям приложения.

б) Регистрация: Будет предусмотрена форма регистрации для новых пользователей с полями для ввода персональных данных, таких как

имя, email и пароль.

в) Просмотр профиля: Пользователи смогут просматривать свой профиль, который будет отображать их персональные данные, а также информацию о бронированиях и истории поиска.

г) Поиск и просмотр жилья: Интерфейс будет включать функцию поиска с возможностью фильтрации по различным параметрам. Пользователи смогут просматривать детали жилища, включая фотографии, описание, цену и отзывы других пользователей.

д) Бронирование жилья: Пользователи смогут легко бронировать выбранное жилье, указывая даты пребывания и количество гостей. Процесс бронирования будет простым и понятным, с четким указанием шагов и подтверждения бронирования.

е) Удаление брони: В случае изменения планов, пользователи смогут отменить свое бронирование с помощью простого и интуитивно понятного процесса.

ж) Просмотр данных о владельце: Пользователи смогут просматривать информацию о владельце жилья, включая контактные данные и отзывы других гостей.

5. При проектировании базы данных определяются ключевые сущности и их взаимосвязи для обеспечения функциональности приложения. В нашем случае основными сущностями являются Пользователи, Токены, Владельцы жилья, Объекты жилья, Цены и Бронирования. Эти сущности были выделены на основе анализа функциональных требований к приложению.
6. В рамках разработки API были определены следующие ключевые эндпоинты, основываясь на функциональных требованиях приложения:

а) **registration/** -Данный эндпоинт используется для регистрации пользователей в системе.

б) **auth/** - Данный эндпоинт используется для авторизации пользователей в системе.

в) **check-code/** - Этот эндпоинт позволяет отправить на указанную пользователю почту код для входа/регистрации.

г) **accommodation/** - Этот эндпоинт позволяет получить информацию о всех объектах жилья.

д) **accommodation-filter/** - Этот эндпоинт позволяет получить инфор-

мацию об отфильтрованных по параметрам (цена, вместимость, колво комнат, тип жилья) объектах жилья.

е) **accommodation-detail/id/** - Этот эндпоинт позволяет получить информацию о конкретном объекте жилья.

ж) **owner-detail/id/** - Этот эндпоинт позволяет получить информацию о конкретном владельце объекта жилья.

з) **owner-accommodation/id/** - Данный эндпоинт позволяет посмотреть все объекты жилья конкретного владельца

и) **booking-date/** - Эндпоинт для работы с бронирования жилья.

к) **cancel-booking-date/id** - Эндпоинт для отмены конкретной брони объекта жилья.

л) **user-detail/** - Эндпоинт для получения информации о пользователе.

м) **user-booking/** - Эндпоинт для получения информации о всех бронях пользователя.

Каждый из этих эндпоинтов поддерживает один или несколько HTTP-методов (GET, POST, DELETE). Кроме того, все запросы к API защищены с помощью JWT-токенов. JWT-токены генерируются при успешной авторизации пользователя и должны быть включены в заголовки всех последующих запросов к API для подтверждения идентификации пользователя.

7. При разработки экранов приложения были реализованы следующие экраны:

а) Экран авторизации, регистрации, ввод кода - при входе в приложение открывается экран авторизации, кнопка "Войти"неактивна, пока пользователь не введет почту нужного формата. При нажатии на кнопку "Регистрация"открывается модальное окно, где необходимо заполнить данные для регистрации. При вводе некорректных данных поля подчеркиваются красным. После нажатия на кнопку "Зарегистрироваться"перебрасывает на экран ввода кода, который пришел на почту. При авторизации также приходит код на электронную почту. Но если указать несуществующий email при авторизации, или ввести неправильный код, который пришел на почту, или при регистрации указать email, на который уже ранее зарегистрировались, то появится ошибка на экрана с информацией о том, какая именно ошибка.

б) Экран пользователя - при нажатии кнопки "Войти"открывается экран профиль пользователя. Где корректно отображается вся информация, которая вводилась при регистрации. Также иконка голубого цвета, если пользователь мужского пола, а если женского - розового. Кнопка "Выйти"активна и при ее нажатии перебрасывает на начальный экран авторизации.

в) Экран поиска жилищ, окно фильтра, экран с информацией о жилье, окно выбора дат бронирования и экран броней пользователя - при переходе на экран "Жилища"появляется список всех доступных жилищ для бронирования с минимальной информацией такой, как тип жилья, цена и адрес. Также доступен фильтр, по которому можно найти жилье подходящее по пожеланию пользователя. При нажатии на карточку жилья, открывается экран с детальной информацией о жилье, где указана дополнительно к краткой информации: описание, колво комнат, кроватей и вместимость, также политика отмены брони и ссылка владельца жилья. Можно посмотреть дополнительные фото жилья, для этого предоставлен удобный слайдер фото. По кнопке "Забронировать"открывается окно, на котором изображен календарь с датами, красный цвет - нельзя забронировать, зеленый - можно. Даты, которые выделены красным нельзя выбрать.

г) Экран броней пользователя - после нажатия кнопки "Забронировать"жилье бронируется и на экране "Брони"появляется информация о брони, также в эти даты уже нельзя забронировать никому, пока пользователь не передумает и не удалит свою бронь. При нажатии кнопки "Удалить бронь"в календаре жилья даты, которые были забронированы пользователем, станут свободны, а на экране "Брони"удалить информация о бронировании.

д) Экран профиля владельца - из экрана детальной информации жилья можно перейти на экран профиля владельца, который сдает это жилье и посмотреть информацию о хозяине и какие жилища он еще сдает.

Заключение.

В ходе данной дипломной работы были изучены и применены современные методы и технологии разработки мобильных приложений на Swift и SwiftUI, а также бекенд-разработки на Python и Django. Целью данного исследования было создание функционального и удобного в использовании приложения для поиска и бронирования жилья.

На основе проведенного анализа и разработки были решены следующие задачи:

- Был проведен подробный обзор и анализ основных принципов и технологий, используемых в разработке мобильных приложений на Swift и SwiftUI. Это включало в себя изучение основных особенностей и преимуществ SwiftUI, сравнение его с аналогами, а также анализ важных сторонних библиотек и архитектурных решений в разработке на Swift.
- Были изучены возможности и особенности разработки бекенда на Python и Django, включая анализ ключевых особенностей и преимуществ Django, сравнение его с аналогами, а также обзор важных сторонних библиотек и архитектурных решений для бекенд-разработки на Django.
- Были определены функциональные и нефункциональные требования к приложению, разработана его архитектура и интерфейс пользователя, проектирована база данных и взаимодействие с бекендом.
- Были реализованы и протестированы мобильное приложение и соответствующий бекенд. Это включало в себя разработку бекенда на Python и разработку iOS приложения на Swift, а также детальное тестирование и отладку приложения.

Результаты исследования и разработки подтвердили, что применение Swift и SwiftUI для разработки мобильных приложений, а также использование Python и Django для разработки бекенда, позволяет создать функциональное, эффективное и безопасное приложение для поиска и бронирования жилья.

Возможными направлениями для дальнейшего развития проекта могут быть оптимизация и улучшение функциональности приложения.