**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(ВлГУ)

Кафедра ВТиСУ

КУРСОВАЯ РАБОТА

по дисциплине «Разработка мобильных приложений»

на тему: «Разработка мобильного приложения на платформе Android. Часть 17. Разработка мобильного приложения для бронирования номеров отеля»

Студент: Тараканов С.В.

Институт: ИИТЭ

Направление: 09.03.03

Группа: ПИ-121

Руководитель: Шутов А. В.

Владимир 2024 г.

**Оглавление**

[Введение 3](#_Toc188521334)

[1. Теоретическая часть 4](#_Toc188521335)

[1.1 Понятие и виды мобильных приложений 4](#_Toc188521336)

[1.2 Описание предметной области и задач на разработку 5](#_Toc188521337)

[2. Практическая часть 7](#_Toc188521338)

[2.1 Анализ и выбор СMS системы для разработки серверной части мобильного приложения 7](#_Toc188521339)

[2.2 Обзор платформ для разработки мобильного приложения 9](#_Toc188521340)

[2.3 Архитектура приложения 12](#_Toc188521341)

[2.4 Обзор разработанного приложения 16](#_Toc188521342)

[Заключение 20](#_Toc188521343)

[Список литературы 21](#_Toc188521344)

# **Введение**

Современный мир стремительно движется в сторону цифровизации, и мобильные технологии занимают ключевую роль в этом процессе. Мобильные приложения стали незаменимым инструментом как для пользователей, так и для бизнеса, предлагая доступ к услугам и продуктам в любое время и из любого места. Одной из наиболее востребованных областей является гостиничный бизнес, где использование мобильных приложений позволяет упростить процесс бронирования и повысить удобство взаимодействия клиентов с отелями.

Целью данной работы является разработка мобильного приложения, предоставляющего пользователям интуитивно понятный интерфейс и функционал для быстрого и удобного бронирования отелей.

Для достижения поставленной цели, необходимо решить следующие задачи:

1. Изучить инструменты и технологии, применимые при разработке приложений на платформе Android.
2. Разработать пользовательский интерфейса с учётом требований удобства и интуитивной понятности.
3. Реализовать функционала приложения.
4. Реализация функционала для просмотра активных заказов.

Таким образом, курсовая работа направлена на приобретение теоретических знаний и практических навыков в области разработки мобильных приложений для Android, что соответствует требованиям образовательной программы и способствует профессиональному развитию в сфере информационных технологий.

# **1. Теоретическая часть**

## **1.1 Понятие и виды мобильных приложений**

Мобильные приложения — это программное обеспечение, специально разработанное для работы на мобильных устройствах, таких как смартфоны и планшеты. Они позволяют пользователям выполнять различные задачи, от общения с другими людьми до получения информации и развлечений.

Мобильные приложения могут быть загружены и установлены на устройства через онлайн-магазины, такие как App Store для iOS и Google Play для Android.

Виды мобильных приложений:

1. Нативные приложения – разрабатываются специально для одной операционной системы (iOS или Android). Используют языки программирования, соответствующие платформе (Swift или Objective-C для iOS, Java или Kotlin для Android).

Обладают высокой производительностью и доступом ко всем функциям

устройства.

1. Гибридные приложения – сочетают элементы нативных и веб

приложений. Разрабатываются с использованием технологий веб-программирования (HTML, CSS, JavaScript) и обертываются в нативный интерфейс.

Позволяют создавать кроссплатформенные приложения, которые работают на разных операционных системах.

1. Веб-приложения – Это сайты, оптимизированные для мобильных устройств и обеспечивающие интерактивный пользовательский опыт. Не требуют установки на устройстве и работают через веб-браузер.

Обновляются на сервере, что позволяет пользователям всегда иметь доступ к последней версии приложения.

1. Прогрессивные веб-приложения (PWA) – это тип веб-приложений, который предлагает функциональность, схожую с нативными приложениями.

Обладают возможностью работы в оффлайн-режиме, отправки уведомлений и установки на устройства. Комбинируют лучшие аспекты веб-приложений и нативных приложений.

1. Игровые приложения – ориентированы на развлечение и взаимодействие с пользователями через игровую механику. Могут быть как нативными, так и гибридными, и часто требуют больших ресурсов для графики и обработки.

Основные черты мобильных приложений:

Доступность: Пользователи могут легко загрузить и установить приложения через магазины приложений.

Интерактивность: Мобильные приложения предлагают интерактивный интерфейс и пользовательский опыт.

Подключение к интернету: Многие приложения требуют постоянного подключения к сети для обновления данных и использования функциональности.

Мобильные приложения становятся все более важной частью нашей жизни, предоставляя пользователям возможности для работы, учебы, отдыха и общения.

## **1.2 Описание предметной области и задач на разработку**

Предметной областью для разработки мобильного приложения является сфера бронирования отелей. Основной целью разработки является создание удобного и интуитивно понятного мобильного приложения, которое позволит пользователям быстро находить и бронировать номера в отелях. Приложение предоставляет возможность просматривать доступные номера с указанием их описания, удобств, стоимости и информации о наличии.

Администратор приложения может управлять данными о номерах, включая добавление, редактирование и удаление информации, а также просматривать сведения о забронированных номерах и управлять заявками пользователей.

Основными пользователями приложения являются клиенты, желающие забронировать отель, и администратор, отвечающий за управление данными.

Задачей проекта является создание мобильного приложения для платформы Android, которое будет обладать следующими функциональными возможностями:

* поиск и фильтрация отелей по параметрам, таким как цена, расположение и удобства.
* просмотр карточек номеров, содержащих их описание, стоимость и фотографии.
* бронирование номеров и управление своими заявками.
* возможности для администратора, включая управление данными о номерах и просмотр забронированных заявок.

# **2. Практическая часть**

## **2.1 Анализ и выбор СMS системы для разработки серверной части мобильного приложения**

Contentful – это облачная headless CMS, которая предоставляет мощные инструменты для создания, управления и доставки контента через API. Она позволяет разработчикам и контент-менеджерам работать отдельно, что значительно упрощает процесс публикации контента.

Преимущества:

1. Позволяет создавать сложные структуры контента с помощью кастомных моделей.

2. Удобный и интуитивный интерфейс для редактирования контента.

3. Поддерживает REST и GraphQL API, что упрощает интеграцию.

4. Обеспечивает хостинг и поддержку, что снижает нагрузку на технический персонал.

Недостатки:

1. Дорогие тарифные планы для масштабируемых проектов.

2. Некоторые ограничения в функционале для работы с большими объемами данных.

3. Невозможно полностью контролировать хостинг и инфраструктуру.

Sanity – это также headless CMS, которая акцентирует внимание на гибкости и кастомизации. Она предлагает уникальные возможности для разработки собственных плагинов и интеграции с другими системами.

Преимущества:

1. Позволяет создавать кастомные типы контента и настройки под конкретные нужды проекта.

2. Поддержка реального времени, что позволяет сразу видеть изменения.

3. Возможность самостийной настройки и расширения системы с помощью Sanity Studio.

4. Легкая интеграция с другими системами благодаря мощным API.

Недостатки:

1. Требует больше времени на первоначальную настройку, особенно для новичков.

2. Может потребоваться время на обучение для команд, не знакомых с инструментом.

Strapi – это open-source headless CMS, которая позволяет разработчикам быстро создать API для управления контентом. Она предлагает средства для создания пользовательских типов контента и имеет встроенную поддержку GraphQL.

Преимущества:

1. Полный контроль над кодом и возможностью хостинга на собственных серверах.

2. Позволяет легко создавать REST и GraphQL API, по сравнению с другими CMS.

3. Удобная панель управления для администраторов и контент-менеджеров.

4. Поддержка плагинов и расширений для кастомизации.

Недостатки:

1. В больших проектах может требовать оптимизации.

2. Требует дополнительных усилий для настройки и развертывания.

3. Меньшая активность сообщества по сравнению с более крупными решениями.

Каждая из этих платформ предлагает уникальные инструменты и подходы к управлению контентом, и выбор между ними зависит от потребностей конкретного проекта и команды.

Для разработки мобильного приложения было принято решение использовать Strapi, по следующим причинам:

1. Strapi — это headless CMS, что означает, что он позволяет вам создавать и управлять контентом без жесткой привязки к конкретному фронтенду. Можно использовать любой фреймворк или библиотеку (React, Vue, Angular и т.д.), что дает большую гибкость в разработке.

2. Strapi автоматически генерирует RESTful и GraphQL API, что позволяет легко интегрировать приложение с другими сервисами и использовать современные подходы к разработке.

3. Strapi поддерживает плагины и механизмы кастомизации, что позволяет разработчикам добавлять новые функции и интегрировать сторонние сервисы согласно специфическим требованиям проекта.

4. Strapi позволяет работать с несколькими базами данных, такими как MongoDB, PostgreSQL, MySQL и SQLite, что позволяет выбрать оптимальное решение для вашего приложения.

5. Strapi предлагает встроенные механизмы управления пользователями и их правами доступа, что обеспечивает защиту данных на уровне API и управления контентом.

6. Strapi позволяет разрабатывать приложения как для веба, так и для мобильных устройств, что делает Strapi идеальным решением для многоплатформенной разработки.

В то время как Contentful и Sanity предлагают свои преимущества, такие как облачные решения и интеграции, Strapi выделяется своей гибкостью, открытостью и возможностью самохостинга, что делает его привлекательным выбором для разработчиков, ищущих контроль и настройку под свои нужды.

**2.2 Обзор платформ для разработки мобильного приложения**

Для разработки приложения на Android существует огромное множество платформ, рассмотрим некоторые из них:

React Native — это популярная платформа с открытым исходным кодом, разработанная Facebook, которая позволяет разработчикам создавать нативные приложения с использованием JavaScript и React. Благодаря обширному сообществу и постоянным обновлениям React Native стал одной из наиболее широко используемых платформ для разработки кроссплатформенных приложений.

Ключевые особенности React Native включают в себя:

* Единая кодовая база для приложений Android и iOS.
* Использование собственных компонентов для повышения производительности взаимодействия с пользователем.
* Горячая перезагрузка для быстрой разработки и тестирования приложений.
* Интеграция с популярными библиотеками и инструментами React.

React Native доступен для Windows, macOS и Linux, и его можно использовать бесплатно.

Недостатки написания приложения на React Native:

* Может потребоваться написание нативного кода для некоторых сложных задач.
* Производительность немного ниже, чем у нативных приложений или Flutter в некоторых сценариях.

Xamarin, продукт Microsoft, представляет собой кроссплатформенную среду разработки приложений, использующую C# для создания нативных приложений для Android, iOS и Windows. Xamarin позволяет разработчикам совместно использовать код на разных платформах, сохраняя при этом возможность доступа к собственным API и функциям для повышения производительности.

Ключевые особенности Xamarin включают в себя:

* Общая база кода C# для приложений Android, iOS и Windows.
* Интеграция Visual Studio для беспрепятственного опыта разработки.
* Доступ к встроенному API для повышения производительности и возможностей приложений.
* Поддержка популярных библиотек и инструментов .NET.

Xamarin доступен для Windows и macOS и бесплатен для проектов с открытым исходным кодом и небольших команд.

Недостатки написания приложения на Xamarin:

* Большой размер приложения из-за дополнительных библиотек.
* Меньшее сообщество по сравнению с React Native и Flutter, что может затруднить поиск помощи.

Flutter — это пакет разработки пользовательского интерфейса с открытым исходным кодом, созданный Google, который позволяет создавать скомпилированные в собственном коде высокопроизводительные приложения для Android, iOS, Интернета и настольных компьютеров с единой кодовой базой. Flutter, созданный на языке программирования Dart, предлагает отличную производительность и множество готовых виджетов для быстрого и бесперебойного процесса разработки.

Особенности Flutter включают в себя:

* Единая кодовая база для Android, iOS, Интернета и настольных компьютеров.
* Быстрая разработка с горячей перезагрузкой и богатым набором готовых виджетов.
* Производительность, подобная нативной, благодаря заранее скомпилированному коду Dart.
* Настраиваемые и расширяемые компоненты пользовательского интерфейса с помощью пакетов.

Flutter доступен для Windows, macOS и Linux, и его можно использовать бесплатно.

Недостатки написания приложения на Flutter:

* Относительно новый фреймворк, что приводит к меньшему количеству доступных библиотек, особенно для специфичных задач.
* Меньшая поддержка для некоторых платформ по сравнению с нативной разработкой.

С учетом всех перечисленных факторов и характеристик, React Native представляется наиболее удобной платформой для разработки мобильного приложения бронирования номеров, так как он будет отлично интегрироваться с выбранной crm системой Strapi.

Это связано с множеством готовых библиотек, а также возможностью быстрой разработки и легкой интеграции с различными API. JavaScript, как язык программирования, обеспечивает гибкость и позволяет разработчикам быстро находить решения для различных задач.

Таким образом, React Native предлагает оптимальный баланс между производительностью и удобством разработки для приложения.

## **2.3 Архитектура приложения**

1. Общая архитектура приложения

Архитектура мобильного приложения для бронирования номеров отеля разработана с использованием современных технологий, которые обеспечивают высокую производительность, удобство разработки и масштабируемость. Она включает серверную часть, построенную на Node.js с использованием CMS Strapi, и клиентскую часть, взаимодействующую с сервером через REST API.

**Клиентская сторона (мобильное приложение):**

Мобильное приложение на React Native предоставляет интуитивно понятный интерфейс и удобное взаимодействие с пользователем. Его архитектура включает несколько ключевых компонентов:

1. Пользовательский интерфейс (UI):

• Состоит из различных экранов и компонентов (например, экран каталога отелей, бронирования).

• Обеспечивает удобное взаимодействие пользователя с приложением.

2. Модель (Model):

• Обрабатывает данные приложения, включая объекты, пользователей и бронирования.

• Отвечает за управление данными, полученными от сервера.

3. Контроллер (Controller):

• Компоненты React Native (например, функциональные компоненты или хуки) управляют логикой взаимодействия между UI и моделью.

• Реализуют обработку пользовательских действий, таких как выбор отеля или отправка заявки на бронирование.

4. Сетевой слой:

• Для взаимодействия с сервером используется библиотека Axios.

• Сетевой слой обрабатывает асинхронные HTTP-запросы, обеспечивая получение данных с сервера без блокировки интерфейса.

• Поддерживает методы GET (для получения данных) и POST (для отправки данных).

**Серверная сторона (Node.js с CMS Strapi):**

1. CMS Strapi:

• Используется для управления контентом приложения, включая отели, пользователей и бронирования.

• Автоматически генерирует REST API, который позволяет клиенту взаимодействовать с сервером.

2. REST API:

• Поддерживает CRUD-операции:

• GET: Получение списка отелей или информации о пользователе.

• POST: Оформление бронирования или регистрация нового пользователя.

• PUT: Обновление информации об отеле или бронировании.

• DELETE: Удаление бронирования.

• Возвращает данные в формате JSON, обеспечивая удобное взаимодействие с клиентом.

3. База данных (SQLite):

• Хранит данные о отелях, пользователях и бронированиях.

• SQLite интегрирована с CMS Strapi для автоматического управления данными.

• Легкая и производительная, что делает её подходящей для небольших и средних приложений.

1. Взаимодействие компонентов

1. UI:

• Пользователь взаимодействует с интерфейсом приложения, выполняя действия, как выбор отеля или оформление бронирования.

• Приложение формирует HTTP-запрос, который отправляется на сервер.

2. HTTP-запрос:

• Используя библиотеку Axios, приложение отправляет запросы на сервер.

• Запросы могут быть:

• GET: Получение данных.

• POST: Отправка данных (например, информация о бронировании).

1. Обработка сервером:

• Сервер на Node.js с помощью CMS Strapi обрабатывает запросы и взаимодействует с базой данных SQLite.

• Выполняются операции выборки, вставки или обновления данных.

1. База данных (SQLite):

База данных, используемая в приложении, реализована на SQLite, что обеспечивает компактное и производительное хранение данных. Однако SQLite не имеет графической оболочки для управления данными, и все операции, такие как создание таблиц, выполнение запросов и управление записями, выполняются через команды SQL. Это требует использования соответствующих инструментов для доступа и работы с базой данных, например, через программный интерфейс или консольные утилиты. Несмотря на отсутствие визуального интерфейса, SQLite остаётся эффективным решением для приложений, благодаря своей лёгкости и интеграции с серверной и клиентской частями.

• Сервер обращается к SQLite для выполнения операций.

• Например, сохранение информации о новом бронировании или обновление данных о доступных отелях.

1. Ответ на клиент:

• Сервер формирует ответ в формате JSON и отправляет его обратно на клиент.

• Пример: список отелей или подтверждение успешного бронирования.

1. Обновление интерфейса:

• Приложение разбирает полученный JSON-ответ и обновляет UI, чтобы отобразить актуальную информацию пользователю.

**Преимущества архитектуры:**

1. Модульность и масштабируемость:

• Использование CMS Strapi упрощает управление контентом и API.

2. Удобство разработки:

• React Native и SQLite обеспечивают лёгкость интеграции и высокую производительность.

3. Гибкость:

• REST API позволяет адаптировать функционал под новые требования.

4. Кроссплатформенность:

• React Native позволяет использовать приложение как на Android, так и на iOS.

База данных не имеет графиической оболочки

Эта архитектура обеспечивает надежность, высокую производительность и удобство как для пользователей, так и для администраторов туристического агентства.

## **2.4 Обзор разработанного приложения**

При открытии приложения пользователю отображается главный экран с перечнем доступных отелей для бронирования. Основной экран содержит следующие элементы:

1. Список отелей

2. Нижняя панель навигации

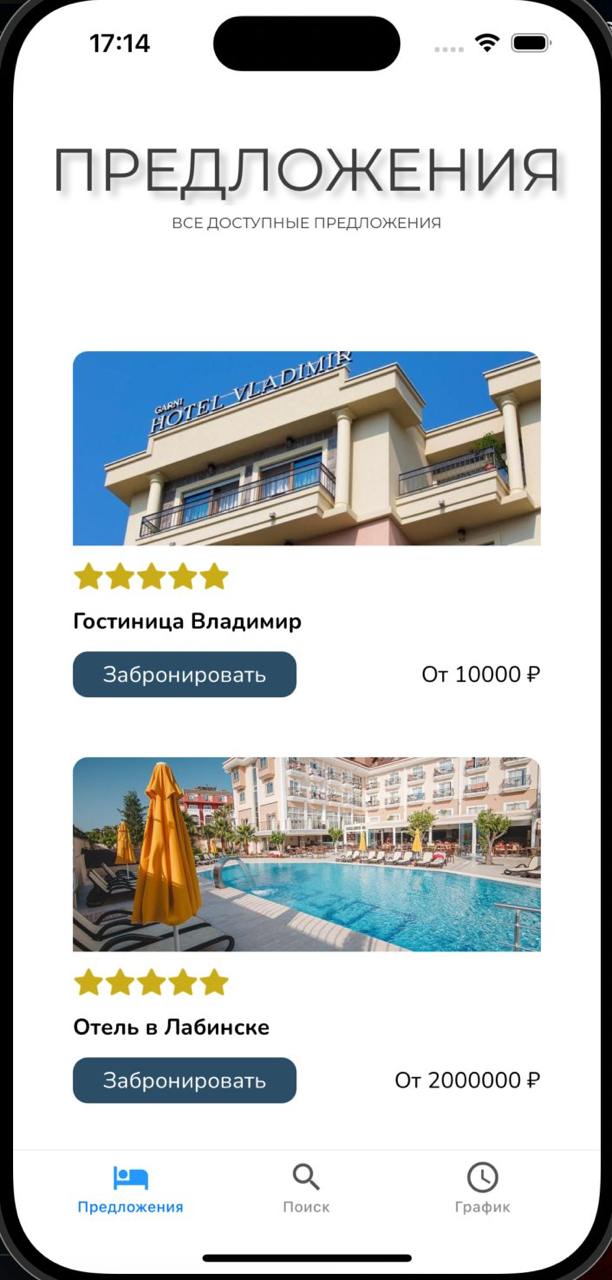


Рисунок 1 – Главная страница

Управление контентом происходит через Strapi «content tipe» – по сути это те же таблицы БД, в которых определяются какие поля будут у сущности, связи и т.п., затем после определения связей можно добавлять эти сущности и выводить в приложении.

Можно выбрать отель из представленного списка и забронировать его (рис. 1-2).

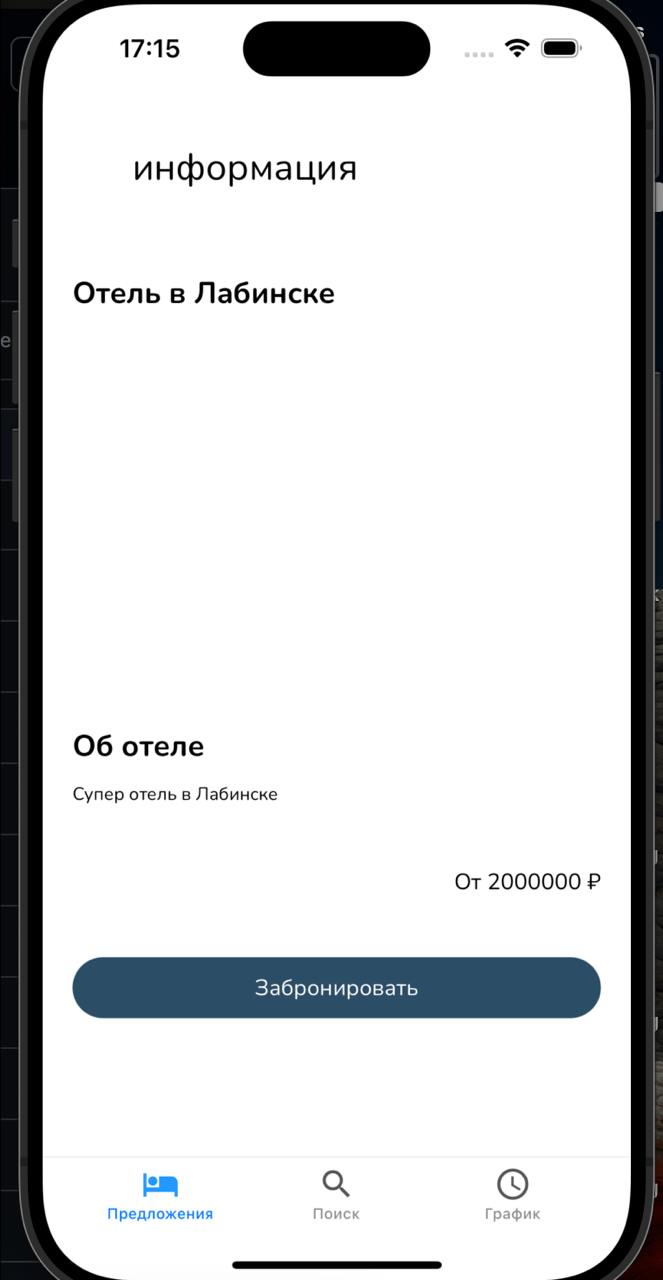


Рисунок 2 – Экран бронирования номера

Экран поиска предназначен для быстрого подбора подходящих вариантов отелей с учетом пользовательских предпочтений (рис.3).

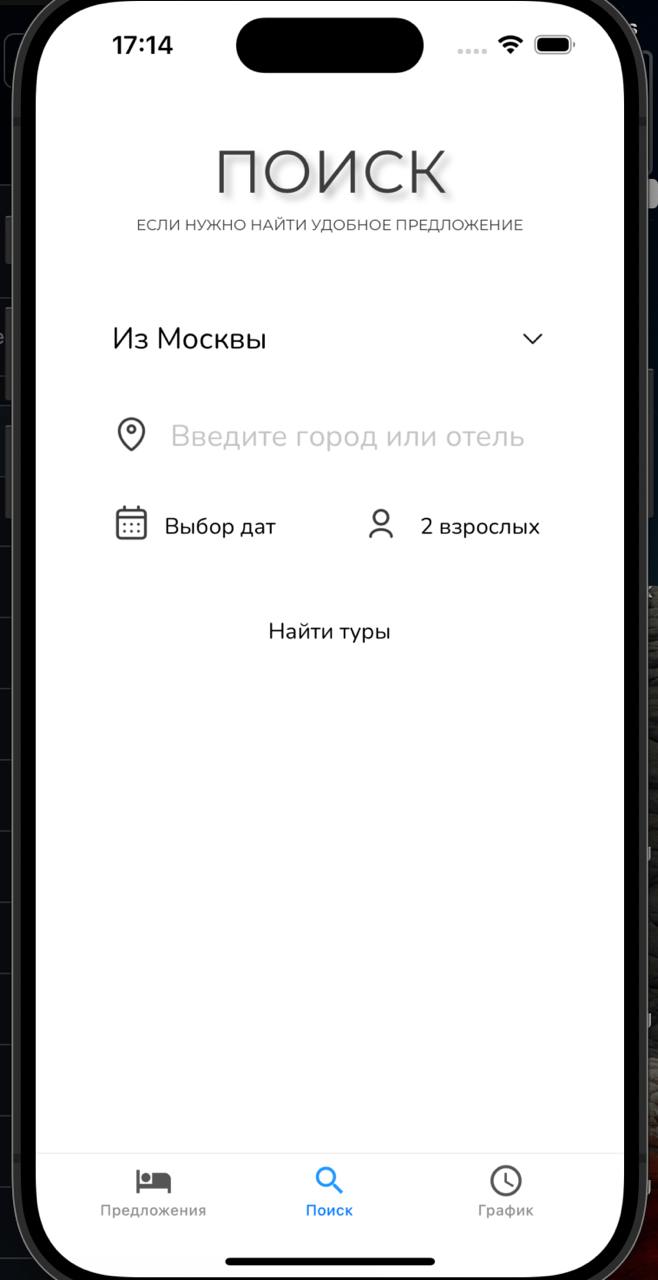


Рисунок 3 – Экран поиска

Экран график представляет пользователю информация о днях недели и времени работы (рис. 4).

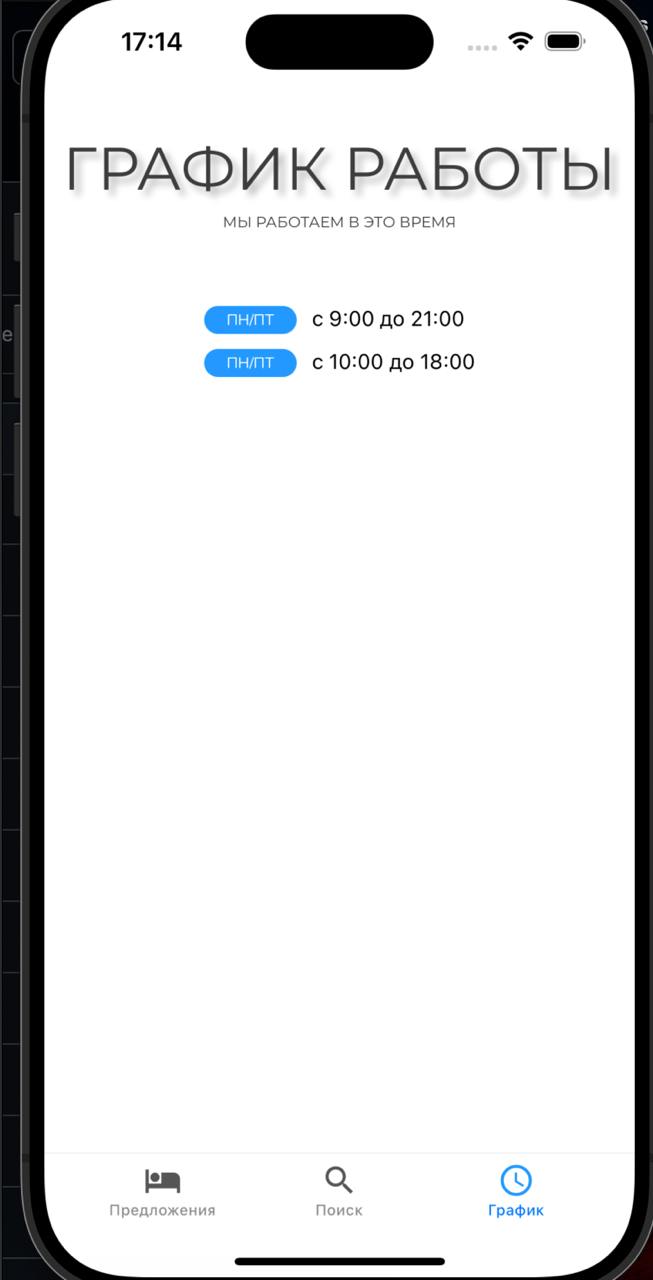


Рисунок 4 – Экран график

# **Заключение**

В процессе разработки мобильного приложения для бронирования номеров отеля была выполнена полная реализация проекта: от постановки задач и анализа требований до создания работающего программного продукта. Приложение включает все ключевые функции, необходимые для удобного бронирования номеров.

Достигнутые результаты:

* Анализ предметной области и существующих решений:
* Проведено исследование современных тенденций в цифровом туризме, что позволило выделить основные требования к приложению.
* Учет актуальных трендов обеспечил создание продукта, соответствующего ожиданиям пользователей.

Проектирование и реализация приложения:

* Разработаны интерфейсы для работы с каталогом отелей, поиском и системой бронирования.
* Интегрирована административная панель для управления турами и обработкой заявок.

Функциональные особенности:

* Пользователи могут просматривать каталог отелей и бронировать туры.
* Администраторы имеют возможность управлять ассортиментом отелей и следить за статусами бронирований.

Практическая значимость:

* Приложение помогает повысить эффективность работы, улучшить взаимодействие с клиентами и увеличить продажи.

# **Список литературы**

1. Шимкин, Д. А. Основы программирования на Java для Android. — М.: БХВ-Петербург, 2021. — 280 с.
2. Фрэнк, М. Мобильные приложения для Android: от идеи до реализации. — М.: Основание, 2020. — 256 с.
3. Android Developers. Документация по Android [Электронный ресурс]. — Режим доступа: https://developer.android.com/docs. — Дата обращения: 15.11.2023.
4. Закутин, В. Б. Проектирование пользовательского интерфейса в мобильных приложениях. — М.: Наука, 2021. — 215 с.
5. Кузнецов, А. В. Лучшие практики разработки Android-приложений. — М.: Альпина Паблишер, 2023. — 320 с.