МИНОБРНАУКИ РОССИИ

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

Кафедра теории функций и стохастического анализа

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ НА OS ANDROID

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

Студентки 4 курса 451 группы
направления 38.03.05 — Бизнес-информатика

механико-математического факультета
Солодченковой Виктории Владимировны

Научный руководитель
доцент, к. э. н. _____ А. Р. Файзлиев

Заведующий кафедрой
д. ф.-м. н., доцент _____ С. П. Сидоров

Саратов 2023

Тема практики:«Проектирование и разработка мобильного приложения на OS Android»

ВВЕДЕНИЕ

Актуальность. По оценкам экспертов, около 75% людей в западных странах, которым необходима психологическая помощь, ее не получают. В России ситуация обстоит еще хуже: по данным ВЦИОМ, только 1% россиян в трудных жизненных ситуациях обращается к психологу. Психотерапия по-прежнему остается для большинства жителей нашей планеты элементом роскоши или социальным запретом. К сожалению, большое количество людей, нуждающихся в помощи, боятся госпитализации, даже если она не грозит, а также опасаются возможной огласки проблемы.

Мобильные приложения в области сохранения психического здоровья появились не так давно, однако многие из них доказали свою эффективность в преодолении сложных ситуаций.

Данная работа представляет интерес, поскольку новые способы проверить здоровье и решить актуальную проблему — это очень большой шаг на пути к осознанному развитию личности.

Целью бакалаврской работы является проектирование и разработка мобильного приложение с функционалом отслеживания своем психологического состояние через личный дневник, наличия научных статей и тестов на тему психология.

Объект исследования – минимальная психологическая помощь в рамках самостоятельного изучения.

Предмет исследования – мобильное приложение для самостоятельной психологической помощи «Инсайт».

Для достижения поставленных целей в работе необходимо решить следующие **задачи**:

1. изучить особенности разработки приложений для ОС Android;
2. спроектировать мобильное приложение с помощью UX-UI-дизайна;
3. определить требования к программе;
4. спроектировать архитектуру разрабатываемого приложения;
5. разработать серверную часть;
6. разработать мобильное приложение.

Практическая значимость в реализации данного мобильного при-

ложения состоит в том, что благодаря «карманному психологу» люди будут более психологически грамотны, смогут больше уделять внимания своему состоянию, что поможет преодолеть сложные жизненные ситуации или предотвратить такие психологические проблемы, как выгорание, апатия и тд. Однако, данное приложение несет лишь информативный характер и не заменяет профессиональную психологическую помощь.

ОСНОВНОЕ СОДЕРЖАНИЕ РАБОТЫ

Во **введении** обосновывается актуальность темы работы, формулируется цель работы и решаемые задачи, отмечается практическая значимость полученных результатов.

В **первом** разделе приводится анализ предметной области, включающий в себя: обзор операционных систем; обзор существующих средств разработки для платформы Android; описание языка программирования Java; обзор технологий для создания распределенного приложения (концепция REST, технология WebSocket).

Во **втором** разделе осуществляется проектирование мобильного приложения. Проектирование включает в себя реализацию UX/UI-дизайна.

UX/UI-дизайн – это проектирование удобных, понятных и эстетичных пользовательских интерфейсов. UX-дизайн включает в себя навигацию по приложению, состав функций внутри цифрового продукта и понятный текст. UI-дизайн – это наполнение приложения, систематизация элементов, выбор цветов, построение визуальной композиции, оформление кнопок, колонок и других графических элементов.

В первую очередь проводился UX-дизайн, а именно анализ конкурентов, проведение опроса возможной целевой аудитории. На данном этапе выявлялось, какие функционалы следует использовать в приложение, а от каких стоит отказаться. Весь функционал будущего приложения описывался в информационной архитектуре.

Информационная архитектура (IA или ИА) – показывает отображенный функционал будущего интерфейса, связи между функциями или экранами, а также зависимость переходов в соответствии с выполненными действиями. Создается она с помощью блок-схем.

Также во время исследований можно проводить более глубокий анализ и составление пользовательской персоны. Пользовательская персона (User Persona) – это вымышленный персонаж, созданный для представления типа пользователя, который может использовать мобильное приложение аналогичных образом. Фокус на отдельном человеке или небольшом сегменте аудитории повышает эмпатию к пользователям, для которых проектируется дизайн. А также проще проверять, все ли потребности были перекрыты после тестирования приложения. Пользовательская персона составляется на основание опросов и интервью с целевой аудиторией, которое проводилось ранее.

По полученным результатам можно ориентироваться в процессе создания приложения, чтобы понимать в то ли направление движемся.

После полученных аналитических результатов переходим к визуальным – мудборду (moodboard), а именно составлению концепции будущего приложения. Мудборд (англ. moodboard – «палитра настроения») – визуальное представление дизайн-проекта, которое состоит из изображений, цветовой палитры и пр. При создании мудборда акцентируется внимание на интересные акцентные элементы в приложение и визуально понятный интерфейс. Это делается на основе других приложений, не обязательно одинакового направления.

Следующий этап создания интерфейса приложения – прототип.

Прототип необходим для:

— сосредоточения на функционале, а не на графическом оформление;
— предотвращения логических ошибок в интерфейсе.

Основная задача прототипа – быстро перевести информационную архитектуру и всю информацию, которая была получена при исследованиях в понятные схематичные экраны. В процессе отрисовке прототипа зачастую происходят изменения в информационной архитектуре.

В процессе было отрисовано около 100 экранов будущего приложения.

По результатам проведений исследований были выявлены следующие функциональные требования:

— система должна предоставлять пользователю возможность регистрации и авторизации;

— система должна позволять пользователю просматривать без подключения к Интернету заранее загруженную информацию;
— система должна предоставлять пользователю возможность менять свои регистрационные данные;
— система должна предоставлять пользователю информацию по разделам «Статьи» и «Тесты»;
— система должна предоставлять пользователю возможность добавлять в «Избранное» любую статью;
— система должна предоставлять просмотр истории всех пройденных тестов пользователем в разделе «Пройденные тесты»;
— система должна предоставлять пользователю большие возможности фильтрации в поиске статей и тестов;
— система должна предоставлять пользователю уведомления о выходе новых статей и тестов;
— система должна предоставлять пользователю возможность добавления новой текстовой записи, редактирования и удаления в разделе «Мой дневник».

Любое Android – приложение состоит из нескольких активностей, которые иногда связаны между друг другом.

Активити - это один экран пользовательского интерфейса приложения, определенная комбинация XMLфайла и Java-файла. По сути, Активити - это контейнер, который содержит как дизайн, так и кодирование. XML-файл представляет нам дизайн нашего экрана. Файл Java содержит код страницы, который отвечает за всё происходящее в активити. Обычно одна из активностей в приложении обозначается как «основная», предлагаемая пользователю при первом запуске приложения. В моем дипломном проекте такой активностью является авторизация пользователей.

После проектирования интерфейса следует визуальная составляющая приложения. Основная задача на данном этапе заключается в том, чтобы привести дизайн к единому стилю.

При составлении визуальной концепции приожения продумываются следующие этапы:

1. Цвета – достаточно использовать 4-5 цветов для создания аккуратного

интерфейса и расставления акцентов.

2. Типографика: регистр, шрифт, длина строки.

При проектирование интерфейса мобильного приложения «Инсайт» использовались следующие цвета: черный, белый, серый, оранжевый, сиреневый, а также следующую типографику: шрифт Roboto с использованием толщины – самый тонкий, нормальный и полужирный.

Третий раздел посвящен архитектуре мобильного приложения. Мобильное приложение состоит из двух основных частей: базы данных и программных компонентов. В данном разделе описывается база данных приложения, программные компоненты, REST-сервис и WebSocket-сервер.

База данных состоит из девяти таблиц: users, profiles, articles, tests, test_questions, test_options, test_results, diary_entries, notifications.

Архитектура приложения будет использовать паттерн проектирования Model-View-Presenter. Основная идея данного паттерна заключается в том, чтобы разделить модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные. Данный паттерн предполагает наличие 3 основных типов классов:

1. Model – классы, отвечающие за хранение данных приложения, а также методы работы с этими данными;
2. View – классы, отвечающие за интерфейс приложения. Они отображает данные, которые передаются классами слоя Presenter, а также реагируют на действия пользователя (например, нажатие кнопки), передавая управление классу слоя Presenter;
3. Presenter – классы, обеспечивающие связь между Model и View. Реагируют на изменения в Model, а также обрабатывают действия пользователя, которые им передали классы слоя View.

REST-сервис обрабатывает http-запросы клиента. Служба REST действует как внутренний сервер, который предоставляет конечные точки RESTful для аутентификации, регистрации пользователей, тестового извлечения и других соответствующих операций. Он взаимодействует с базой данных для хранения и извлечения данных, таких как пользовательская информация и

тестовые данные.

Сочетание REST-сервиса с базой данных и WebSocket-сервером обеспечивает гибкую и масштабируемую архитектуру. Служба REST обрабатывает традиционную связь запрос-ответ, в то время как сервер WebSocket обеспечивает управляемую событиями связь в режиме реального времени между сервером и клиентом. Эта настройка позволяет мобильному приложению взаимодействовать с серверной системой с помощью REST API, а также получать обновления в режиме реального времени через WebSocket-соединения, когда это необходимо.

Четвертый раздел посвящен разработке мобильного приложения.

Для реализации взаимодействия с базой данных было решено использовать специальную ORM-библиотеку. ORM (англ. Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных». Цель: работа с данными в терминах классов, а не таблиц. Для этого была выбрана библиотека Hibernate.

Для реализации REST-сервиса используется платформа Spring. Платформа Spring – популярная платформа приложений с открытым кодом, предназначенная для упрощения разработки для J2EE. Она состоит из контейнера, платформы управления элементами и набора интегрируемых служб для веб-интерфейсов пользователя, транзакций и сохранения состояния. В состав платформы Spring входит Spring Web – расширяемая платформа MVC для создания веб-приложений. Spring MVC построен вокруг центрального сервлета (DispatcherServlet), который распределяет запросы по контроллерам, а также предоставляет другие возможности при разработке веб приложений. Сервлет (Servlet) – это java-программы, которые выполняются на серверной стороне Web-приложения.

Класс с настройками для Spring Security получает доступ к базе данных (mDataSource) и позволяет пользователям аутентифицироваться по идентификатору (который хранится в клиентском приложении) и паролю. В базе данных на сервере пароли хранятся в зашифрованном виде (для шифрования используется алгоритм bcrypt), поэтому, создается и объект класса PasswordEncoder для шифрования приходящего пароля и последующей про-

верки, которая определена в методе configureGlobal. В методе configure определяются адреса, к которым ограничивает доступ система защиты.

Для создания Socket-сервера используется WebSocket API в Java EE 7 для обеспечения связи между клиентом и сервером в реальном времени.

Для создания WebSocket-сервера необходимо создать класс и указать для него аннотацию: @ServerEndpoint и указать URI в качестве параметра. Эта аннотация показывает, что данный класс является классомтерминалом и может принимать сообщения по протоколу WebSocket В WebSocket-сервере реализованы методы onMessage(String message, Session session), onOpen(Session session) и onClose(Session session), а также метод getQueryMap для обработки запроса при подключении нового клиента. При подключении нового клиента к серверу, вызывается метод onOpen, который добавляет новую сессию в список сессий mSessions. После этого, при получении сообщений от клиента, вызывается метод onMessage. В нем входящее сообщение обрабатывается специальным объектом mMessageManager класса MessageManager. Этот класс принимает JSON- сообщения от клиентов и отправляет ответы в зависимости от полученного сообщения. Метод onClose вызывается при завершении сеанса: он удаляет текущую сессию из списка сессий mSessions.

Далее проводилась реализация компонентов Model-View-Presenter.

Компонент Presenter состоит из:

1. AuthPresenter – отвечает за авторизацию;
2. RegistrationPresenter – отвечает за регистрацию;
3. MainActivity – основная активность приложения;
4. HomeActivity – отвечает за главную страницу;
5. AppCompatActivity – отвечает за перемещение по разделам;
6. TestActivity – отвечает отображение содержимого выбранного теста;
7. DiaryPresenter – отвечает за страницу с личным дневником;
8. ArticlesPresenter – отвечает за страницу с статьями;
9. ProfilePresenter - обеспечивает отображение данных пользователя;
10. TestsPresenter – отвечает за основную страницу с тестами;
11. FinishTestPresenter - отвечает за сохранение результатов теста и отправления полученного опыта на сервер;
12. TestListPresenter - обеспечивает отображение доступных тестов для вы-

бранной коллекции.

Компонент Model состоит из:

1. UserModel - представляет пользовательские данные;
2. UserRepository - определяет контракт для операций с данными, относящихся к пользователю;
3. DiaryEntry - представляет собой запись в дневнике;
4. Article - данные статей;
5. UserProfile – данные профиля пользователя;
6. TestModel – данные о тесте;
7. TestRepository - определяет контракт для операций с данными, связанных с тестами.

Компонент View состоит из:

1. AuthActivity - экран авторизации;
2. RegistrationActivity – экран регистрации;
3. ProfileContract.View – экран профиля;
4. ArticlesContract.View – экран статей;
5. RecyclerView – экран отображения списка статей;
6. TestListView – экран тестов;
7. TestsContract.View – экран тестов по выбранным параметрам;
8. DiaryContract.View – экран личного дневника;
9. RecyclerView – экран со списком записей в дневнике.

В результате была выполнена реализация всех компонентов системы в соответствии со всеми требованиями к системе. В том числе были реализованы мобильное приложение для ОС Android и сервер на языке программирования Java.

ОСНОВНЫЕ РЕЗУЛЬТАТЫ

Целью данной работы являлась разработка мобильного приложения для самостоятельной психологической помощи путем изучения научных статей, прохождения психометрических тестов и отслеживания своего состояние через личный дневник.

В ходе работы были выполнены следующие задачи:

— изучены особенности разработки приложений для ОС Android;

— спроектировано мобильное приложение с помощью UX/UI-дизайна;
— определены требования к программе;
— спроектирована архитектуру разрабатываемого приложения;
— разработана серверная часть;
— разработано мобильное приложение.