1 Сессия

**Общее**

Необходимо разработать мобильное приложение для смартфона, удовлетворяющее следующим требованиям:

Минимальная версия ОС, поддерживаемая приложением, должна быть: Android - 13.0, iOS - 14.0.

**В работе необходимо использовать систему контроля версий Git, которую предоставляет организатор.**

Все проекты проверяются с Git репозиториев. Необходимо загрузить каждую сессию в отдельную ветку с именем «Session\_Х», где Х – это номер сессии. По завершению каждой сессии необходимо делать средствами GitHub Merge слияние с основной веткой, которая должна называться «Main», при этом ветка удаляться не должна.

В качестве бэкенда будет использован Supabase. Авторизация в нем будет происходить через выданный аккаунт GitHub.

**Необходимо строго следовать предложенному дизайну.**

**Вся верстка должна быть адаптивной (следует учитывать разные размеры экранов). Необходимо:**

* **Избегать появления большого пустого пространства;**
* **Следить за отсутствием искажения элементов;**
* **Все элементы должны полностью находится в границах и на месте, указанном в макете;**
* **Учитывать расстояние между элементами;**
* **Используйте шрифты согласно макету.**

**Дизайн предложен в Figma:**

<https://www.figma.com/file/MluNlGZBfxE45DUUlJEEdx/OECH-APP?type=design&node-id=14-7747&mode=design&t=mEbo48F3pMKRk1S0-0>

Необходимо осуществлять комментирование кода в созданных классах. Обязательны следующие комментарии:

* Описание назначения класса
* Дата создания
* Автор создания
* Описание назначения вложенных элементов программного кода

При разработке проекта приложения вам необходимо использовать архитектуру ([см.файл с описанием архитектуры](https://docs.google.com/document/d/1DHGJzhbW0XN6D1P-USjFZkrPoljXKINbpMohblCme8I/edit?usp=sharing)), в которой будут разделены слои бизнес-логики, представлений и домена. Изменение бизнес-логики и/или представления одного из экранов не должно повлечь за собой изменение других экранов и нарушение работоспособности приложения, за исключением переходов. Допускается использование Supabase.

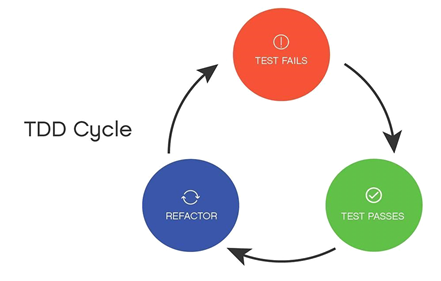
Файлы проекта распределены по папкам в соответствии с архитектурой. Допустимо использование папки Common для общих файлов.

Необходимо корректно обрабатывать запросы к серверу. В случае получения ошибки от сервера или отсутствия соединения с сетью Интернет необходимо отобразить соответствующий текст ошибки в диалоговом окне, которое должно закрываться только пользователем. В процессе обмена данными с сервером должна осуществляться стандартная индикация.

Все медиа ресурсы должны кэшироваться.

**Сессия 1**

Необходимо реализовать следующий функционал, следуя методологии TDD:



1. Создайте проект. Настройте иконку приложения согласно макету.

2. Создайте SplashScreen, как на макете.

3. Реализуйте приветственные экраны (Onboarding), как на макете.

Логика экрана:

У Вас есть определенная последовательность изображений и текста. Это значит, что у Вас должна быть очередь (queue) этих изображений и текстов. При свайпе влево или вправо необходимо извлекать по одному изображению и тексту и показывать на экране до тех пор, пока изображения и текст в очереди не закончатся.

4. Экран «Onboarding»:

* Данные очереди не должны храниться в компоненте (в компоненте хранятся только данные текущего элемента).
* Очередь создаётся единственный раз при первом запуске и не пересоздаётся.
* Сохраняется прогресс отображения Onboarding (при перезапуске должен отобразиться следующий набор изображения и текста, на котором остановился пользователь).
* При нажатии на «Sign in» осуществить переход на экран «Log in», при нажатии на кнопку «Sing Up» осуществить переход на экран «Sing up» и сохранить информацию о том, что пользователь уже смотрел Onboarding (не показывать ее при последующих запусках), а также очистить очередь.

5. Создайте экран «Sign Up», как на макете:

* Реализовать просмотр политики конфиденциальности, которая хранится локально в проекте в виде PDF файла и открывается в свободной форме.
* Регистрация и переход на экран «Home» осуществляется при нажатии на кнопку «Sign Up», только при согласии с Условиями и политикой конфиденциальности и заполнении всех полей.
* Реализуйте отправку запроса на сервер для регистрации.

6. Создайте экран «Log in», как на макете:

* При нажатии на «Forgot Password» осуществляется переход на экран «Forgot Password».
* Когда все поля ввода заполнены, при нажатии на кнопку «Log in» осуществляется переход на экран «Home».
* Реализуйте отправку запроса на сервер для авторизации с помощью почты и пароля.

7. На экранах «Sign Up» и «Log in»:

* Реализуйте проверку e-mail на корректность (соответствие паттерну «name@domenname.ru», где имя и доменное имя может состоять только из маленьких букв и цифр). При некорректном заполнении необходимо отобразить ошибку любым способом.
* Реализуйте возможность регистрации посредством использования активного аккаунта Google.
* При нажатии на кнопку «Sing Up» осуществляется переход на экран «Sign up», а при нажатии на «Sign in» осуществляется переход на экран «Log in».
* Обеспечить безопасное хранение пароля, используя SHA-512.
* Реализуйте возможность отображения пароля.

8. Создайте необходимые классы для последующего тестирования. Добавьте объявления методов, реализовывать их не нужно.

* RED - Реализуйте тесты (UI или модульные), проверяющие поведение приложения:
* Изображение и текста из очереди извлекается правильно (в порядке добавления в очередь).
* Корректное извлечение элементов из очереди (количество элементов в очереди уменьшается на единицу).
* В случае, когда в очереди несколько картинок, устанавливается соответствующий индикатор (точки навигации).
* Если очередь пустая появляется кнопка «Sing Up» и при нажатии на «Sing in», происходит открытие пустого экрана «Home» приложения. Если очередь не пустая – переход отсутствует.

Минимальная структура тестов следующая:

* объект класса, который мы тестируем;
* корректные/некорректные входные данные;
* сравнение полученного результата с ожидаемым (Assert).

Сделайте коммит с текущим состоянием тестов (Failed) и оставьте сообщение RED.

GREEN - напишите реализацию логики так, чтобы все тесты проходили успешно (в случае изменения теста на данном этапе, тест не будет учитываться в оценке).

Сделайте коммит с текущим состоянием тестов (Succes) и оставьте сообщение GREEN.

REFACTOR: проведите рефакторинг не менее трех различных участков кода, созданного на предыдущем этапе (в случае изменения теста на данном этапе, тест не будет учитываться в оценке). Рефа́кторинг (англ. refactoring), или перепроектирование кода, переработка кода, равносильное преобразование алгоритмов - процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

Сделайте коммит с текущим состоянием тестов (Succes) и оставьте сообщение REFACTOR.