1 Сессия

**Общее**

Необходимо разработать мобильное приложение для смартфона, удовлетворяющее следующим требованиям:

Минимальная версия ОС, поддерживаемая приложением, должна быть: Android - 11.0, iOS - 14.0.

В качестве бэкенда будет использован Supabase. Для авторизации в supabase используйте учетную запись, **выданную главным экспертом.**

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

**Необходимо строго следовать предложенному дизайну.**

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), 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.

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

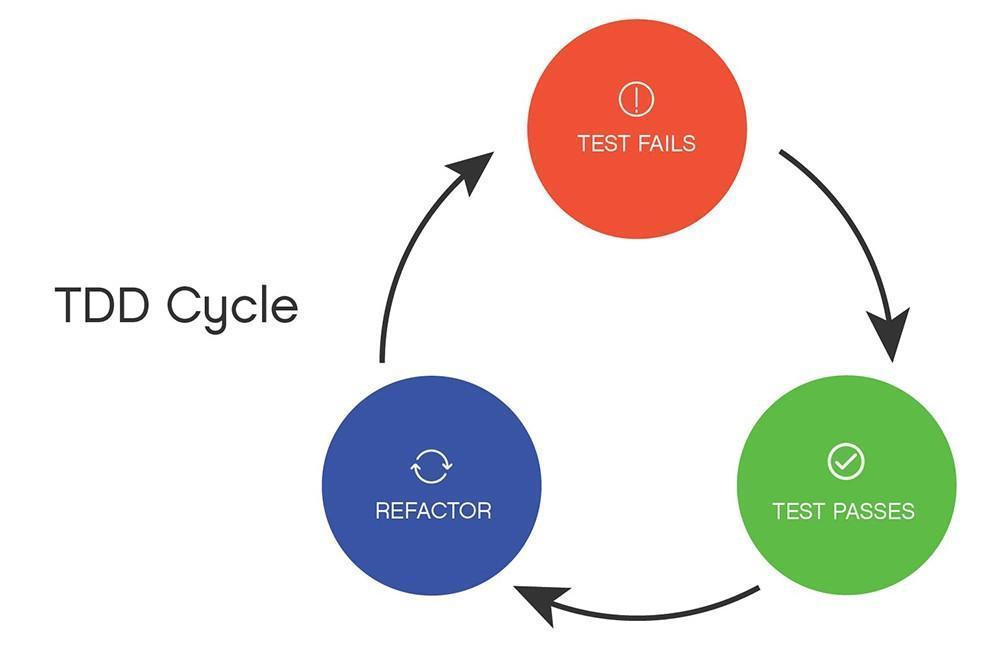
* **Избегать появления большого пустого пространства;**
* **Следить за отсутствием искажения элементов;**
* **Все элементы должны полностью находится в границах и на месте, указанном в макете;**
* **Учитывать расстояние между элементами;**
* **Используйте шрифты согласно макету.**
* **Дизайн предложен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

[**https://www.figma.com/file/guyXW190UglGSwE2CCyafu/OECH-APP-Final?type=design&node-id=0-1&mode=design&t=lrg58KLyrOO3iQa5-0**](https://www.figma.com/file/guyXW190UglGSwE2CCyafu/OECH-APP-Final?type=design&node-id=0-1&mode=design&t=lrg58KLyrOO3iQa5-0)

Установите иконку приложения, согласно макету

Создайте SplashScreen, как на макете

Реализуйте приветственные экраны (Onboard) согласно методологии TDD и макету:



Логика экрана:

* + У Вас есть определенная последовательность изображений и текста. Это значит, что у Вас должна быть очередь (queue) этих изображений и текстов. При нажатии на кнопку «Next» необходимо извлекать по одному изображению и тексту и показывать на экране до тех пор, пока изображения и текст в очереди не закончатся. Изображения и текст при переключении должны сопровождаться анимацией проявления (исчезает текущий и появляется следующий);
  + Когда в очереди останется последнее изображение и текст — нужно поменять кнопки;
  + Когда пользователь увидит последнее изображение и нажмет кнопку "Sing Up", нужно показать пустой экран «Holder». Учесть это, и сделать Holder, а не Home
  + При нажатии на кнопку «Skip», осуществить переход на экран «Holder» приложения и сохранить информацию о том, что пользователь уже смотрел Onboard (не показывать ее при последующих запусках), а также очистить очередь!.

1. Необходимо во время сессии работать в ветке “Path-X”, где Х – это номер сессии. По завершению сессии необходимо сделать средствами Giltab Merge, с основной веткой, которая должна называться “main”, при этом ветка удалятся не должна.
2. Экран «Onboard»:
   * 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 Onboard (При перезапуске должен отобразиться следующий набор изображения и текста, на котором остановился пользователь);

Создайте необходимые классы для последующего тестирования. Добавьте объявления методов, реализовывать их 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 правильная надпись на кнопке.
    - Случай, когда очередь пустая, надпись на кнопке должна измениться на "Sing Up".
    - Если очередь пустая и пользователь нажал на кнопку “Sing in”, происходит открытие пустого экрана «Holder» 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.