Лабораторная работа № 10. Проектирование мобильного приложения с организацией непрерывной сборки и автоматического тестирования

Выполнение лабораторной работы включает решение следующих задач:

1. Проектирование интерфейса приложения и его дизайн.
2. Разработка требований, включая диаграммы вариантов использования, диаграммы классов, диаграммы последовательности, диаграммы пакетов, диаграммы развертывания (deployment diagram) и физическую модель базы данных.
3. Описание требований, включая features, user stories, tasks, bugs, и планирование выполнения работ в github project или trello или иная система.
4. Использование системы контроля версиями git, репозиторий для группы
5. Разработка unit- и UI-тестов
6. Автоматизация сборки проекта и настройка сервиса непрерывной интеграции Github Actions.
7. Разработка мобильного приложения на swift и интерфейса приложения с помощью UIKit или SwiftUI.

Содержание

Темы проектов

Группе выбрать одну из тем или предложить собственное мобильное приложение, которое будет реализовано на языке swift на основе UIKit или SwiftUI.

Проект 1. Разработка мобильного приложения для банка: клиент запускает приложение, вводит логин / пароль и переходит в меню. На первом этапе он может выбрать действие “Просмотреть счета”. После чего перейдет на экран с отображением всех его счетов, как активных, так и заблокированных (закрытые счета не отображаются). Поддерживаются типы счетов: текущий, сберегательный, кредитный и карт-счет. Карт-счет может быть зарплатным, сберегательным и кредитным. Для зарплатных счетов поддерживается овердрафт. Из дополнительных функций обеспечивает просмотр курса валют, отображение карты с выводом филиалов отделений и поиск ближайшего отделения по отношению к местоположению клиента. Данные должны храниться в базе данных sqlite. Использовать UserDefaults (NSUserDefaults) или plist.

Проект 2. Требуется разработать приложение бронирования отеля. Пользователь выбирает отель, бронирует, подтверждает бронирование. После подтверждения бронирования ему доступен интерфейс выбора вида транспорта (поезд, автобус, самолет) и выбор маршрута на карте. После выбора маршрута ему предлагается список рейсов с классификацией по видам транспорта. Содержится название рейса (компании) и стоимость проезда/перелёта. Сервис поддерживает регистрацию. Данные регистрации могут храниться локально или же может использоваться внешний сервис. Дополнительный функционал приветствуется. Данные должны храниться в базе данных sqlite. Использовать UserDefaults (NSUserDefaults) или plist.

Проект 3. Требуется разработать приложение для сервиса заказа еды. Сервис заказа еды может быть агрегатором нескольких ресторанов. Пользователь выбирает вид блюда, может изменять его характеристики, подтверждает заказ и выбирает способ оплаты (оплата онлайн, ЕРИП, терминал или наличные), а так же указывает адрес доставки. Пользователю должен быть доступен интерфейс карты для выбора ближайшего ресторана или предлагается автоматически ближайший, из выбранной категории ресторанов при определении местоположения пользователя. Пользователю должен быть доступен комментарий к заказу для детализации или уточнения. Сервис поддерживает регистрацию. Данные регистрации могут храниться локально или же может использоваться внешний сервис. Дополнительный функционал приветствуется. Данные должны храниться в базе данных sqlite. Использовать UserDefaults (NSUserDefaults) или plist.

Проект 4. Требуется разработать приложение для заказа талонов в поликлинику. Сервис должен быть агрегатором поликлиник города. Поддерживает регистрацию пользователя. При регистрации определяется ближайшая поликлиника (детская и взрослая) или же пользователь может выбрать поликлинику из списка. Так же должен быть доступен перечень больниц города и их отделений, включая их специализацию. Для поликлиник обеспечить заказ талона к педиатру/терапевту или узкому специалисту, а так же возможность отмены или переноса талона на другую дату. Данные регистрации могут храниться локально или же может использоваться внешний сервис. Дополнительный функционал приветствуется. Данные должны храниться в базе данных sqlite. Использовать UserDefaults (NSUserDefaults) или plist.

Проект 5. Требуется разработать приложение для заказа товаров, например печатных и электронных книг, а так же канцтоваров и/или других товаров. Пользователь может выбирать товары, изменять их количество в заказе, подтверждает заказ и выбирает способ оплаты, а так же указывает адрес доставки. Так же должен быть доступен функционал отмены и разделения заказа, в случае отсутствия одного из товаров. Требуется реализовать отмену оплаты или переноса оплаты на другой заказ, если оплата выполнена онлайн. Пользователю должен быть доступен интерфейс карты для выбора ближайшего пункта самовывоза или предлагается автоматически ближайший согласно местоположению пользователя. Сервис поддерживает регистрацию. Данные регистрации могут храниться локально или же может использоваться внешний сервис. Дополнительный функционал приветствуется. Данные должны храниться в базе данных sqlite. Использовать UserDefaults (NSUserDefaults) или plist.

Проект 6. Свой проект, в котором используется навигация по карте, вывод изображений, поиск, обработка жестов, авторизация пользователя. Данные должны храниться в базе данных sqlite. Использовать UserDefaults (NSUserDefaults) или plist.

Задание № 1. Создание проекта для управления разработкой

Данный проект выполняется в группе из 2-3 участников (максимально 4 участников). Создать новый проект в стиле Kanban, используя github projects или https://trello.com. Добавить в него членов команды, распределив роли следующим образом: *Менеджер проекта, Разработчик, Проектировщик интерфейсов, Тестировщик.*

Задание № 2. Проектирование интерфейса приложения, разработка требований и документирование проекта

Упражнение 2.1. Проектирование интерфейса приложения

Разработать mockup-ы (дизайн-макеты) приложения и добавить в проект высокоуровневые требования (features) и user stories, распределив работы между участниками команды. Дизайн-макет приложения может быть разработан с помощью приложения онлайн-сервиса Figma или других (Uxpin, Invision). Перед подготовкой макета познакомиться с руководством по проектированию интерфейсов для iOS — iOS Human Interface Guidelines.

Работы распределить таким образом, что спецификацию готовят менеджер проекта и разработчик, а проектировщик интерфейса — разрабатывает дизайн-страницы (mock-up) приложения. Требуется разработать как минимум 3 вида экрана приложения. Например, если у нас банковское приложение, то разрабатываем экран авторизации, экран (dashboard) после авторизации и еще один экран, например, просмотра списка счетов.

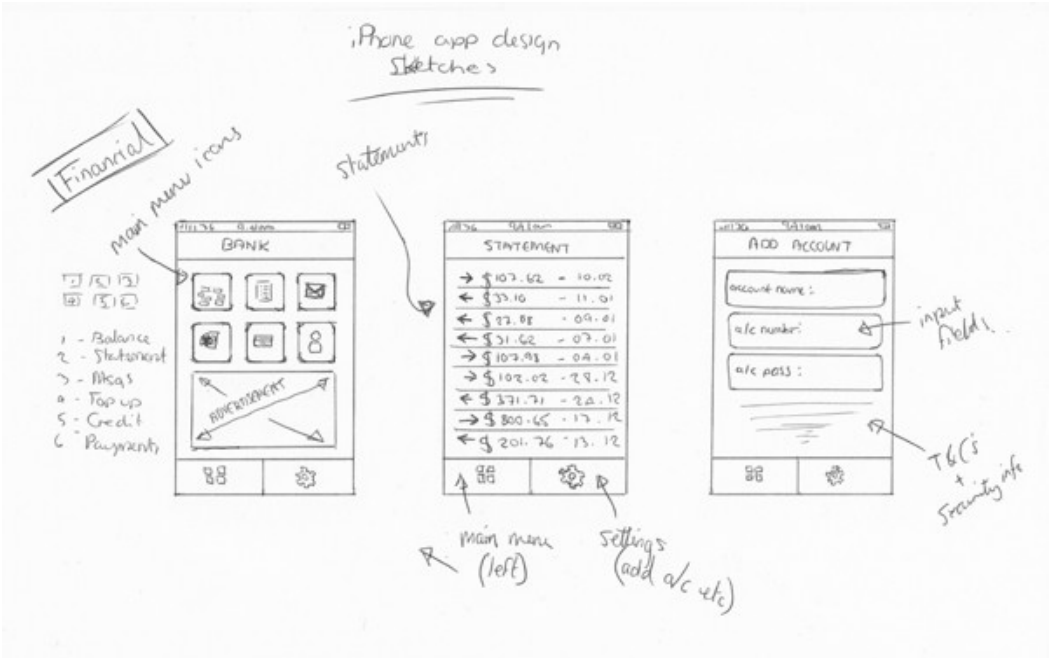

Рис. 1. Пример эскиза (макета) приложения

Упражнение 2.2. Разработка требований

Разработать спецификацию требований согласно структуре ниже:

1. Назначение приложения/системы
2. Общее описание (пользователи, требования к программному и аппаратному

обеспечению)
3 Спецификация требований
  3.1 Функциональные требования
  3.2 Удобство использования
  3.3 Требования надёжности
  3.4 Требования производительности
4 Диаграммы
  4.1 Варианты использования со сценариями
  4.2 Диаграммы классов
  4.3 Физическая модель БД

Упражнение 2.3. Документирование проекта

Изучить документацию https://guides.github.com/features/wikis/ и https://lab.github.com/githubtraining/github-pages. Документировать проект в Readme, wiki репозитория и в Github pages согласно следующим требованиям ниже. Структура страниц в Github Pages должна быть аналогичной wiki:

1. Файл Readme и wiki оформить с помощью синтаксиса Markdown.
2. Структура файла Readme должна быть следующей:
   - **Project Name**: в данном блоке указать название проекта.
   - **Description**: Краткое описание проекта и его функциональности в 3-5 предложениях.
   - **Installation**: Последовательность шагов, как установить приложение локально.
   - **Usage**: Рекомендации как использовать приложение после установки. Может содержать скриншоты.
   - **Contributing**: Сведения об авторах проекта и какие задачи реализовывали.
3. Структура страниц wiki должна быть следующей:
   - **Главная страница**: содержит краткое описание задачи и ссылки на другие материалы и разделы.
   - **Функциональные требования**: описание функциональных требований, диаграммы Use case, текстовые сценарии.
   - **Диаграмма файлов приложения**: диаграмма файлов и описание.
   - **Дополнительная спецификация**: ограничения, требования к безопасности, надежности и другое.
   - **Схема базы данных**: страница содержит схему базы данных в виде изображения сущностей CoreData.
   - **Презентация проекта:** ссылка на презентацию проекта, в которой должно быть отражено распределение задач в команде, требования к приложению, схема базы данных

Учебные материалы

Ознакомьтесь с учебными материалами по проектированию макетов:

- https://www.figma.com/ — онлайн-сервис прототипирования Figma
- http://figmadesign.ru/uroki-figma.html — уроки по figma
- https://www.youtube.com/channel/UCA_gpio7fvwvMSzHedEgO1w — уроки по figma

Задание № 3. Настройка сервиса непрерывной интеграции и разработка Unit- и UI-тестов

1) Клонировать репозиторий и создать ветки для релиз-версии (production) и

тестовой версии (development). Согласовать правила работы с ветками. При необходимости задействовать механизм запросов на изменение (pull requests).

2) Для автоматизации сборки проекта настроить сервис непрерывной интеграции Github Actions. Настройку выполняет менеджер проекта.

Перед настройкой изучить статьи:

1. Github Actions for iOS projects — https://sarunw.com/posts/github-actions-for-ios-projects/
2. iOS CI/CD with GitHub Actions — https://medium.com/thefork/ios-ci-cd-with-github-actions-e4504228c9d

и пример шаблона настройки сборки https://github.com/Apple-Actions/Example-iOS.

3) Добавить Unit- и UI-тесты, чтобы сервис Github Actions перед сборкой выполнял тесты и формировал сборку только после их прохождения.

Учебные материалы:

Как за день поднять CI для iOS-разработчиков — https://habrahabr.ru/company/livetyping/blog/302128/

QA Automation

QA Automation, часть 1: CI сервер наше все

QA Automation, часть 2: Автотестирование – Начало.

Автоматическое тестирование iOS приложений — http://habrahabr.ru/post/202910/

iOS Unit Testing For Beginners Tutorial With Swift 3 — https://www.youtube.com/watch?v=-VdQfX6q_a8

iOS Unit Testing and UI Testing Tutorial — https://www.raywenderlich.com/150073/ios-unit-testing-and-ui-testing-tutorial

Unit Testing on macOS: Part 1/2 — https://www.raywenderlich.com/141405/unit-testing-macos-part-12

About Testing with Xcode — https://developer.apple.com/library/content/documentation/DeveloperTools/Conceptual/testing_with_xcode/chapters/01-introduction.html#//apple_ref/doc/uid/TP40014132-CH1-SW1

Строим домашний CI/CD при помощи GitHub Actions и Python — https://habr.com/ru/post/476368/

GitHub Actions: Continuous Delivery of Swift Packages — https://medium.com/xcblog/github-actions-continuous-delivery-of-swift-packages-25da4e5ccff6

Github Actions CI for Swift Projects — https://medium.com/rosberryapps/github-actions-ci-for-swift-projects-c129baceed1a

Building an Objective-C or Swift Project — https://docs.travis-ci.com/user/languages/objective-c/

CI for Swift Frameworks — https://kean.github.io/post/ci-for-frameworks

Travis CI Tutorial: Getting Started — https://www.raywenderlich.com/1618-travis-ci-tutorial-getting-started

Задание № 4. Разработка приложения

На основе спецификации, описанной в задании 2, макетов приложения, и функционала на основе user stories реализовать приложение на языке Swift с интерфейсом на основе UIKit или SwiftUI.

- Архитектура приложения должна соответствовать шаблону MVC или может быть выбран иной архитектурный шаблон на выбор: MVP, MVVM или Viper.
- Код приложения должен содержать комментарии в стиле pragma mark. При разработке кода следовать руководству стиля для оформления кода на языке Swift:
  1. Руководство стиля для swift на русском языке — http://ilya2606.ru/?p=1846
  2. https://github.com/RedMadRobot/RMR-swift-style-guide
  3. https://github.com/raywenderlich/swift-style-guide
- Приложение должно быть локализовано как минимум на 3 языка (английский и русский **обязательно**, на **выбор** один из языков белорусский/польский/украинский/французский/немецкий/испанский или иной).
- Для проектирования интерфейса использовать библиотеку SwiftUI (https://developer.apple.com/tutorials/swiftui/tutorials, https://habr.com/ru/company/tinkoff/blog/454750/).
- Дизайн приложения должен соответствовать рекомендациям из Apple Human Interface Guidelines (https://developer.apple.com/design/human-interface-guidelines/)