

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	2
1 Основная часть	3
1.1 Характеристика предприятия	3
1.2 Характеристика проделанной работы	3
1.2.1 Анализ фреймворка UIKit	3
1.2.2 Создание макетов пользовательского интерфейса	6
1.2.3 Реализация пользовательского интерфейса	7
ЗАКЛЮЧЕНИЕ	11
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	12

ВВЕДЕНИЕ

Целью производственной практики является изучение фреймворка UIKit и разработка пользовательских интерфейсов приложений для iOS с его помощью.

Задачи производственной практики:

- провести анализ средств фреймворка UIKit;
- спроектировать слой пользовательского интерфейса (UI);
- разработать слой пользовательского интерфейса (UI).

1 Основная часть

1.1 Характеристика предприятия

ООО «ВЕЛЛХОУМ» предлагает такие услуги, как подбор и монтаж автономной канализации, подбор и установка септиков, включающие в себя полный цикл работ по установке: определение типа грунта, глубины подвода коммуникации, уровня грунтовых вод, геологии участка и выбора модели оборудования.

Ранее созданием и поддержкой сайта компании занималась сторонняя организация. Однако с недавнего времени фирма обзавелась собственным ИТ-отделом, в штате которого бэкенд и фронтенд разработчики, мобильные разработчики для Android и iOS, тестировщики, аналитики и дизайнеры.

Ранее компания не имела мобильного приложения, позволяющего клиентам отслеживать статусы заказов, а также запрашивать работы по обслуживанию уже установленного оборудования. Однако в связи с ростом организации данная возможность оказалась необходима, в связи с чем в штат были приняты iOS и Android разработчики. Команда разработки iOS-приложений специализируется на языке программирования Swift. Макеты приложения создаются при помощи сервиса Figma.

1.2 Характеристика проделанной работы

1.2.1 Анализ фреймворка UIKit

UIKit [1] — фреймворк, предоставляющий архитектуру окон и представлений для реализации пользовательского интерфейса, включая компоненты, которые можно использовать для построения базовой инфраструктуры приложения.

Альтернативой выступает фреймворк SwiftUI [2], предоставляющий не меньше возможностей, однако, будучи представленной в 2016 году, библиотека не смогла стать столь популярной в разработке. Большинство команд, работающих над пользовательскими интерфейсами iOS приложений, отдают предпочтение UIKit.

Базовыми UI-элементами, предлагаемыми фреймворком UIKit, являются UIView и UIViewController.

UIView (или представление) [3] — это фундаментальный блок пользовательского интерфейса приложения, а UIView-класс определяет поведение, общее для всех представлений. Объекты этого класса отображают содержимое в пределах своих границ и обрабатывают любые взаимодействия с этим содержимым. Для отображения надписей, изображений, кно-

пок и других элементов интерфейса, обычно встречающихся в приложениях, используют не определяемые самостоятельно подклассы *view*, а предоставляемые платформой *UIKit*.

UIViewController (или контроллер) [4] — объект, который управляет иерархией представлений приложения. Основные обязанности контроллера включают следующее:

- обновление содержимого представлений;
- реагирование на взаимодействие пользователя с представлениями;
- изменение размеров представлений и управление макетом общего интерфейса;
- координация с другими объектами, включая другие контроллеры представления.

Также фреймворк предоставляет несколько методов верстки интерфейса мобильного приложения. Сравнительный анализ по ключевым критериям методов представлен в таблице 1.1. Для краткости записи в данной таблице используются следующие обозначения критериев:

- К1 — масштабируемость интерфейса;
- К2 — возможность командной разработки;
- К3 — сложность внесения изменений;
- К4 — возможность обработки всех параметров UI-элемента;
- К5 — скорость работы метода.

Таблица 1.1 — Классификация методов верстки интерфейса мобильного приложения

Класс метода	Название метода	К1	К2	К3	К4	К5
Метод ручной верстки	Frame	Нет	Да	Низкая	Да	Высокая
Метод автоматической верстки	Autolayout	Да	Да	Высокая	Да	Средняя
	Interface Builder	Да	Да	Высокая	Нет	Низкая

Более детальное описание характеристики методов верстки представлено в таблице 1.2.

Таблица 1.2 — Результат анализа

Признак классификации	Название метода	Характеристика метода
Ручная верстка	Frame	Метод требует отдельной обработки каждой вариации размеров экрана в связи с заданием константными значениями размеров и координат элементов интерфейса, однако этот же фактор предоставляет возможность разделения задачи верстки на подзадачи (в пределах одного представления) и упрощается процесс внесения изменений в ранее написанный код. Поскольку frame предполагает программную верстку, разработчику открывается возможность оперировать любыми параметрами UI-элемента. Скорость работы метода совпадает со скоростью размещения элементов на экране, так как не требует дополнительных вычислений.
Автоматическая верстка	Autolayout	При грамотно составленных ограничениях, посредством которых элементы располагаются на экране, метод предоставляет возможность создавать интерфейс, масштабируемый для всех типов устройств. Командная разработка возможна в пределах одного родительского представления, внесение изменений в размеры или координаты одного элемента может повлечь за собой внесение изменений и в другие элементы сцены. Поскольку Autolayout предполагает программную верстку, разработчику открывается возможность оперировать любыми параметрами UI-элемента. Время работы метода увеличивается при увеличении взаимосвязей между объектами сцены.
Автоматическая верстка	Interface Builder	При грамотно составленных ограничениях, посредством которых элементы располагаются на экране, метод предоставляет возможность создавать интерфейс, масштабируемый для всех типов устройств. Командная разработка возможна в пределах одного экрана. Верстка предполагает работу с инструментом создания интерфейсов, а не написание кода, поэтому явно проследить правила задания ограничений возможности нет, что усложняет задачу внесения изменений в размеры или координаты, а также редактирование одного элемента может повлечь за собой внесение правок в другие. Interface Builder предоставляет ограниченный набор параметров графического элемента интерфейса, доступных для обработки. Время работы метода увеличивается при увеличении взаимосвязей между объектами сцены, а также зависит от времени преобразования графического элемента в код.

В качестве метода верстки для приложения предприятия был выбран метод AutoLayout в силу своей масштабируемости, возможности командной разработки, а также возможности более легкого внесения изменений в последующем.

1.2.2 Создание макетов пользовательского интерфейса

Функционал приложения

Приложение должно предоставлять пользователю следующий функционал:

- авторизация;
- настройка профиля и привязка банковской карты;
- выбор геопозиции;
- поиск необходимой услуги, а также информации о ней;
- заказ услуги и комплектующих;
- отслеживание статуса заказа;
- получение уведомлений о необходимости обслуживания оборудования.

Работа в Figma

Для создания макетов пользовательского интерфейса приложения использовался сервис Figma [5].

Все экраны приложения показаны на рисунке 1.1.

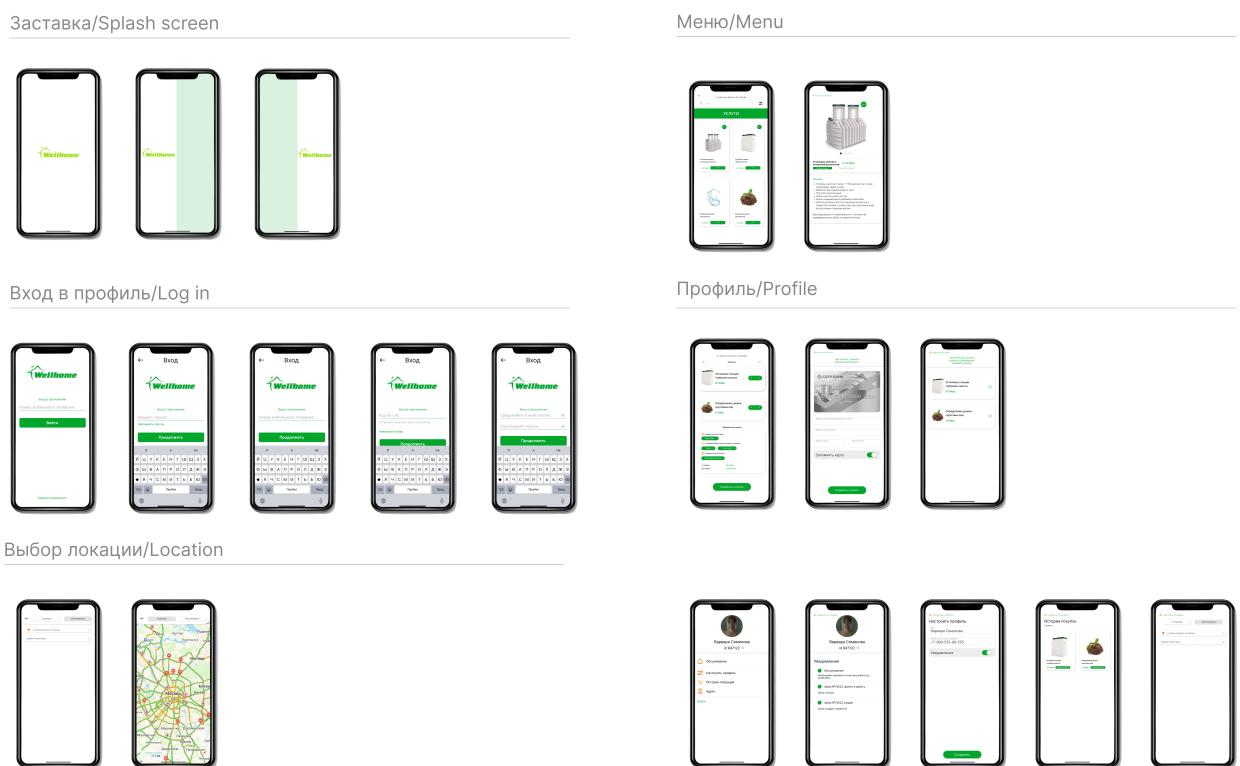


Рисунок 1.1 — Все экраны приложения

1.2.3 Реализация пользовательского интерфейса

Средства разработки

В основе всего приложения заложен архитектурный паттерн, применяемый при создании мобильных приложений, MVP — Model View Presenter. По спроектированным макетам был разработан слой пользовательского интерфейса приложения, который и оказался реализацией слоя View.

В процессе создания приложения использовалась среда разработки XCode [6], предоставляющая весь необходимый функционал для написания кода на языке программирования Swift, реализации паттерна MVP, а самое главное — открывающая возможность тестировать и запускать код на симуляторе мобильного устройства.

Среда разработки и симулятор показаны на рисунке 1.2.

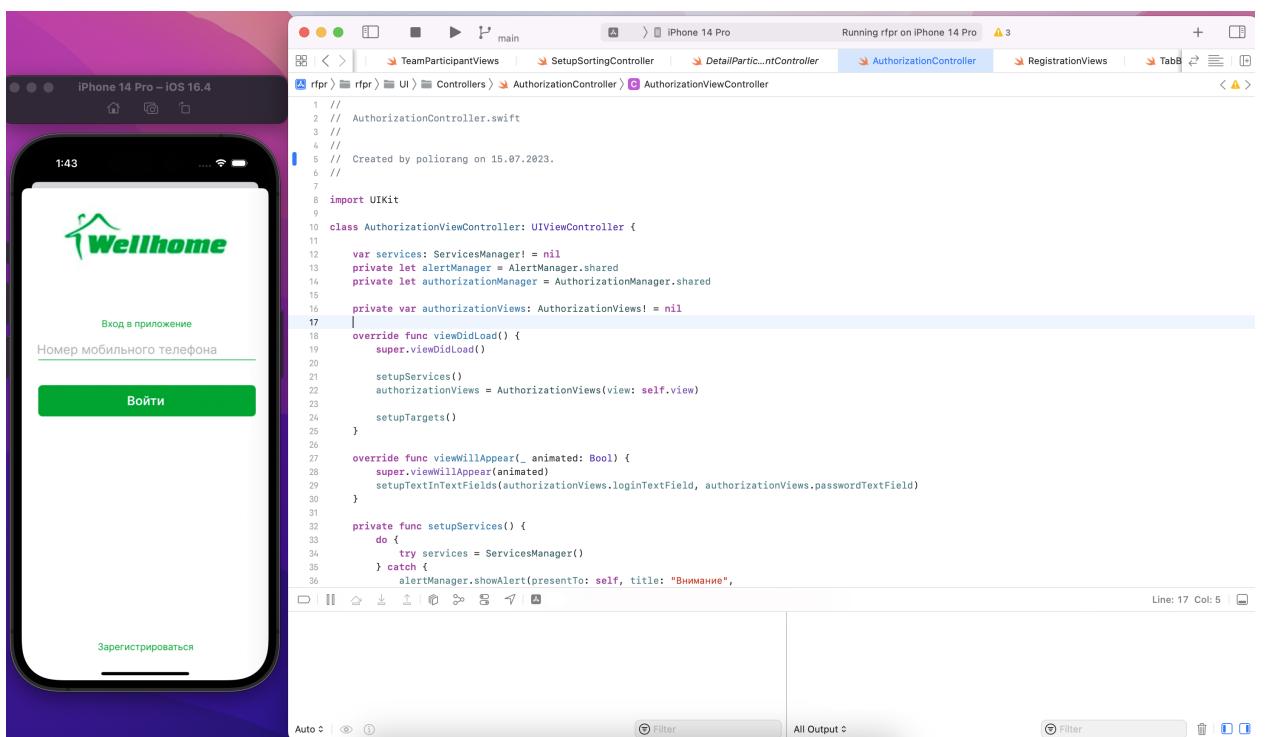


Рисунок 1.2 — Среда разработки и симулятор

Разлизаия

Реализация начального экрана приложения показана на рисунке 1.3.



Рисунок 1.3 — Начальный экран приложения

Реализация экрана авторизации показана на рисунке 1.4.

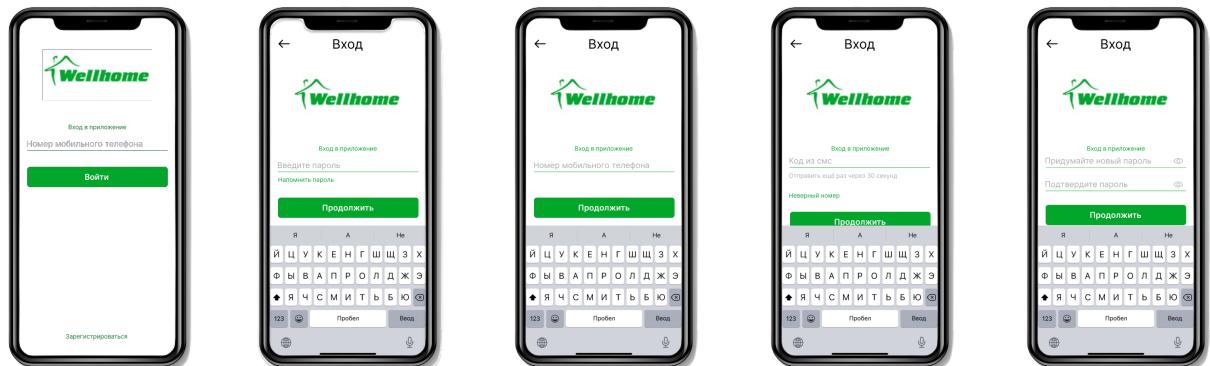


Рисунок 1.4 — Экран авторизации

Реализация экрана выбора локации показана на рисунке 1.5.

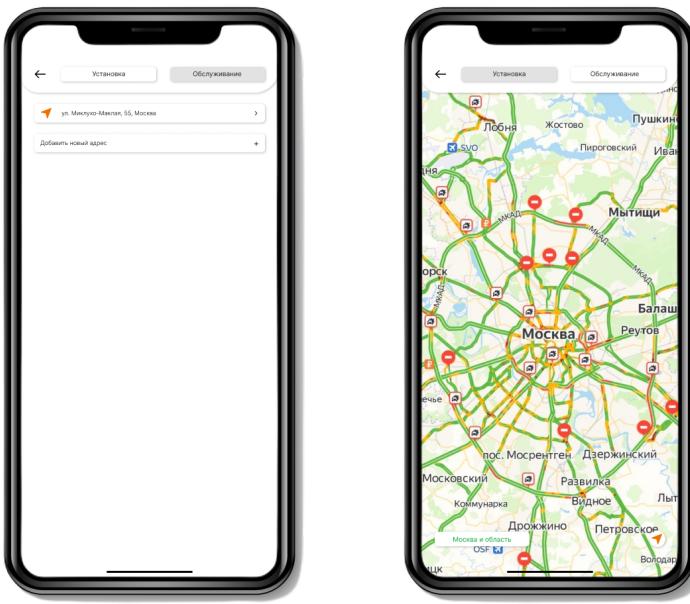


Рисунок 1.5 — Экран выбора локации

Реализация экрана поиска необходимой услуги и информации о ней показана на рисунке 1.6.

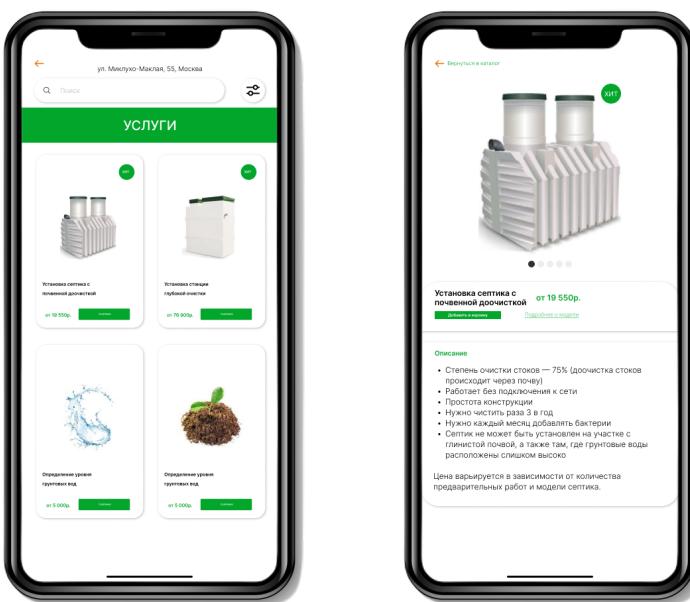


Рисунок 1.6 — Экран поиска услуги

Реализация экрана корзины клиента показана на рисунке 1.7.

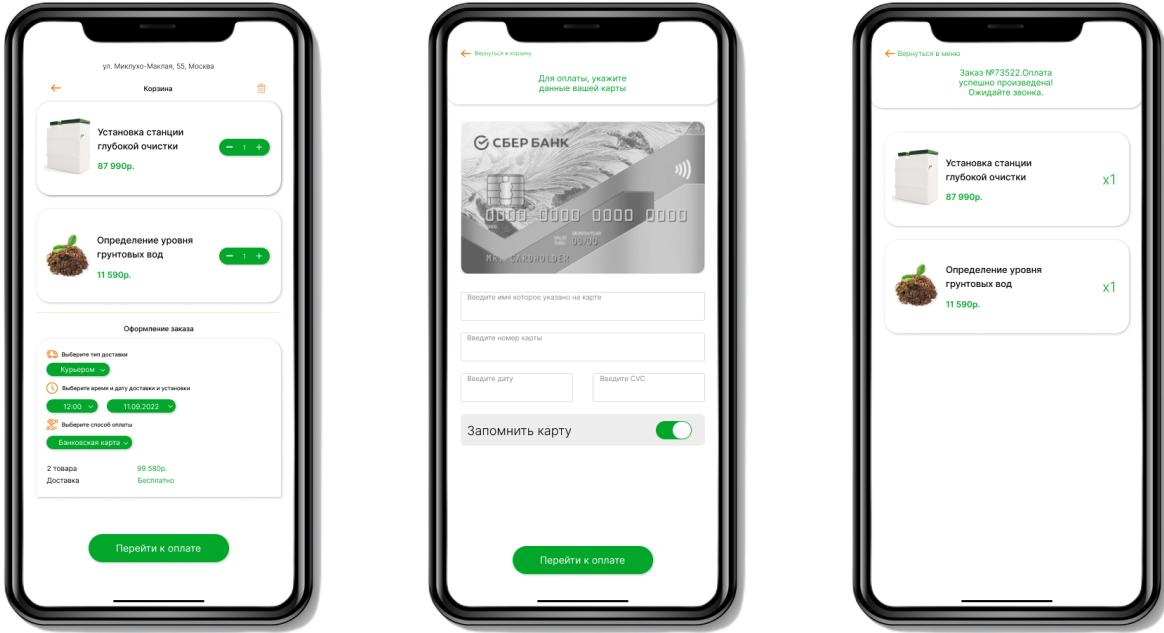


Рисунок 1.7 — Экран корзины клиента

Реализация экрана профиля показана на рисунке 1.8.

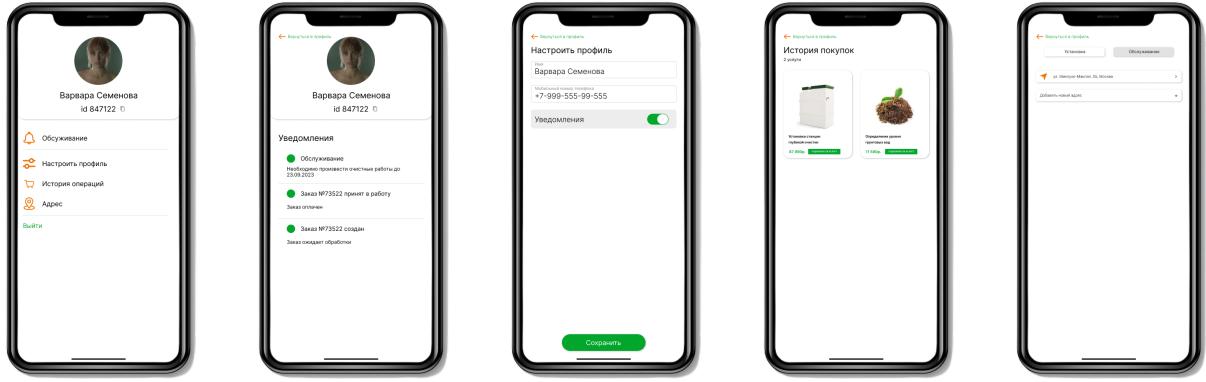


Рисунок 1.8 — Экран профиля

Поскольку при создании интерфейса приложения использовался метод AutoLayout фреймворка UIKit, слой UI масштабируем. Новые сотрудники в команде iOS-разработки смогут быстрее разобраться с уже имеющимся и кодом, чем, например, если бы проект был реализован при помощи InterfaceBuilder. В силу этого и добавление нового функционала будет происходить быстрее.

ЗАКЛЮЧЕНИЕ

В рамках производственной практики был изучен фреймворк UIKit и с его помощью разработан пользовательский интерфейс приложения для iOS. Для достижения этой цели были выполнены следующие задачи:

- проведен анализ средств фреймворка UIKit;
- спроектирован слой пользовательского интерфейса;
- разработать слой пользовательского интерфейса.

Цель производственной практики достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. UIKit | Apple Developer Documentation [Электронный ресурс]. — Режим доступа: <https://developer.apple.com/documentation/uikit/uikit>, свободный (дата обращения: 20.07.2023)
2. SwiftUI | Apple Developer Documentation [Электронный ресурс]. — Режим доступа: <https://developer.apple.com/documentation/swiftui>, свободный (дата обращения: 20.07.2023)
3. UIView | Apple Developer Documentation [Электронный ресурс]. — Режим доступа: <https://developer.apple.com/documentation/uikit/uiview>, свободный (дата обращения: 20.07.2023)
4. UIViewController | Apple Developer Documentation [Электронный ресурс]. — Режим доступа: <https://developer.apple.com/documentation/uikit/uiviewcontroller>, свободный (дата обращения: 20.07.2023)
5. Figma: The Collaborative Interface Design Tool [Электронный ресурс]. — Режим доступа: <https://www.figma.com>, свободный (дата обращения: 21.07.2023)
6. XCode | Apple Developer Documentation [Электронный ресурс]. — Режим доступа: <https://developer.apple.com/xcode/>, свободный (дата обращения: 20.07.2023)