Project headquarters

Система управления и автоматизации процессов взаимодействия с жильцами многоквартирных домов.

Цель

Создание opensource продукта для управления многоквартирными домами (далее МКД).

Проблема

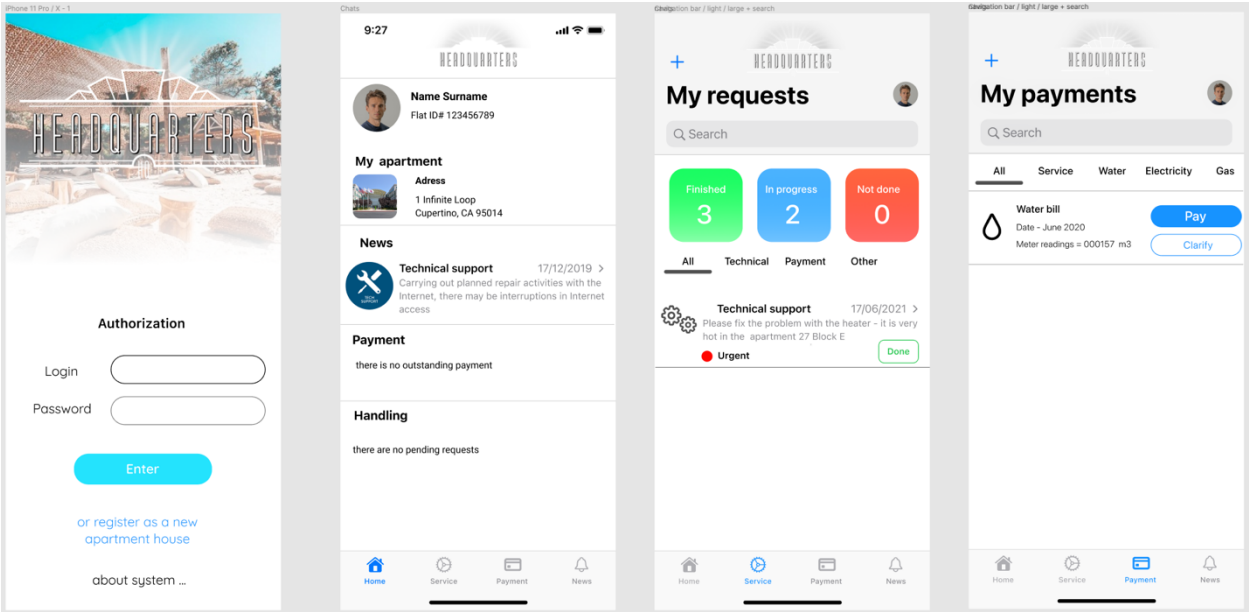
80% населения мира проживает в многоквартирных домах. В процессе жизнедеятельности имеются процессы коммуникаций жителей МКД с управляющими (обслуживающими) компаниями. В большинстве случаев данные процессы протекают длительно, децентрализовано, и конфликтно. В частности, коммуникации по вопросам взаиморасчетов, релевантности данных показателей счетчиков энергоносителей или коммуникации по вопросам технического обслуживания (сантехника уборка и др.)

Сценарии использования

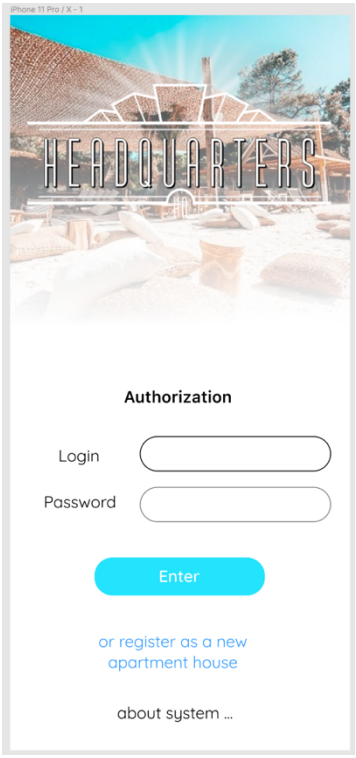
Для клиента:
Управляющая компания устанавливает приложение и добавляет в свою учетную запись своих клиентов – собственников квартир. У каждого пользователя в приложении имеются все данные по их собственности, история по взаиморасчетам, и кабинет запросов на техническое обслуживание, оперативный чат с представителем УК , и канал оперативной информации от управляющей компании. В кабинете управляющей компании появляется единый канал коммуникации и учета.

Для разработчиков:
Создание opensource проекта позволит любому разработчику реализовать множество коммерческих проектов, брать за основу данную систему и допиливать систему под собственную серверную сторону (aws Heroku etc), брендинг под конкретного клиента, добавлять интеграции с поставщиками энергоресурсов и прочее.

Прототипы интерфейсов

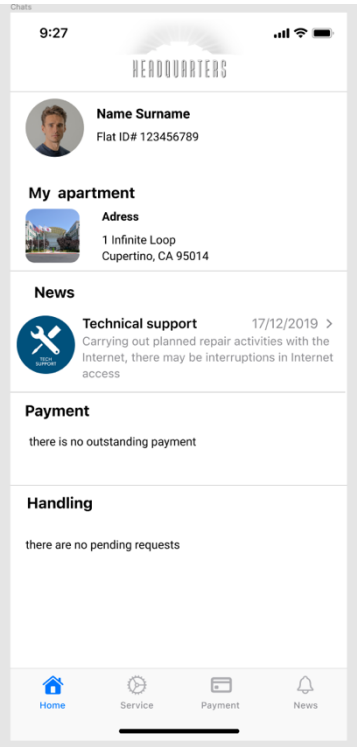


Описание прототипов интерфейсов



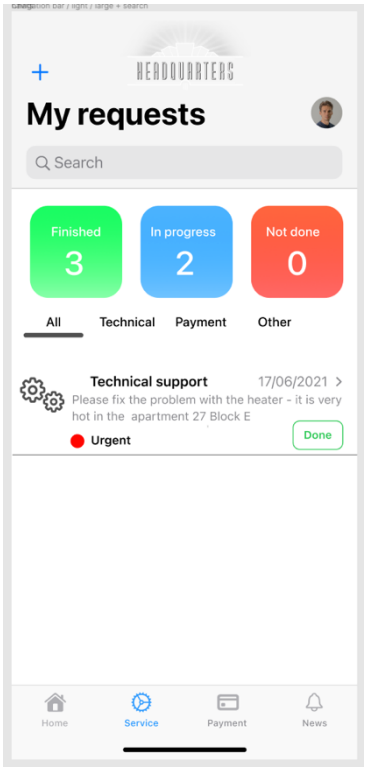
Экран авторизации в систему.

Возможностью регистрации нового кабинета подразумевает SAAS формат, однако на стадии MVP мы данный вариант не рассматриваем, для каждой коробки необходимо разворачивать персональный сервер что реализовать на стороннем сервисе проблематично(долго) либо дорого.



Основной кабинет клиента включает в себя Таббар с 4 пунктами меню

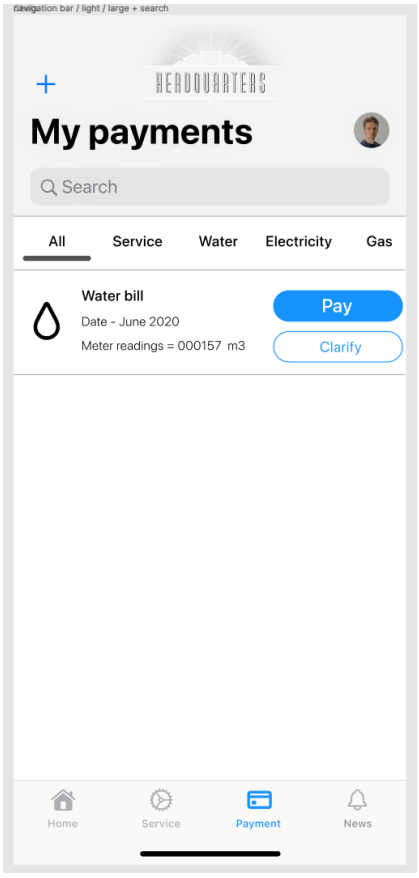
1) Основное меню – основная информация по пользователю включающая сводную информацию по его регистрационным данным, раздел с последними новостями от УК, раздел платежей (последние не оплаченные счета), раздел обращений в сервисную службу (последние тикеты в статусе «В работе» и «На рассмотрении»)
2) Сервис – раздел для создания заявок на сервис и работа с ними
3) Оплата – раздел для оплаты выставляемых счетов
4) Новости – раздел с уведомления от Управляющей компании



Раздел «Сервис»

Включает в себя возможность создания и обработки заявок на обслуживание, предусмотрено 3 вида обращений
- по техническим вопросам
- по вопросам платежей. и расчетов
- прочие вопросы организационного характера

Имеется блок поиска, аналитического блока по выполнению тикетов(для SLA УК), и фильтр по разделам



Раздел «Оплата»

Имеется список счетов на оплату по группам энергоносителей с возможностью оплатить через платежную систему или подать заявку на уточнение (создание тикета в сервис со связкой к выставленному счету).

Предусмотрен блок поиска счетов и фильтр п о разделам:
- Все счета
- Сервисные платежи
- Вода
- Электроэнергия
- Газ

Возможно необходимо предусмотреть систему подсчета общей задолженности. И фильтрацию. По оплачено/не оплачено.

Пользовательские роли

Предусматриваются 2 основные роли пользователей:

1) Администратор – это сотрудники обслуживающей (управляющей) компании, в частности, Управляющий , Бухгалтер, Техгическая служба, Аварийная служба.
2) Пользователь – Собственник недвижимости (житель) который имеет свой персональный кабинет для общения с сервисной службой и бухгалтерией и управляющей компанией по всем основным вопросам.

В дальнейшем возможно реализовать отдельный кабинет для Технического обслуживания – для них оставить только раздел «Сервис» с расширенными возможностями по управлению тикетами.

Структуры данных

Для реализации данной системы необходимо реализовать серверную сторону, для первоначальной системы достаточно подготовить коробочную версию – 1 сервер для одного клиента который можно реализовать на VAPOR и развернуть на Heroku. Или интегрировать AWS сервисы, что предпочтительнее.

Архитектура

Клиентское приложение предлагается реализовать на SwiftUI или UIKit с архитектурой MVVM и подключить AWS сервисы для Push уведомлений и серверную часть. Возможно подключение локальной клиентской базы данных REALM и настроить асинхронную синхронизация с сервером. AWS.

Сроки

Для реализации MVP потребуется 2 месяца и 4 fullstack разработчика.