Описание решения

Содержание

[Общее описание 1](#_Toc497207523)

[База данных 2](#_Toc497207524)

[Серверная часть 3](#_Toc497207525)

[Клиентская часть 4](#_Toc497207526)

[Тестирование 6](#_Toc497207528)

# Общее описание

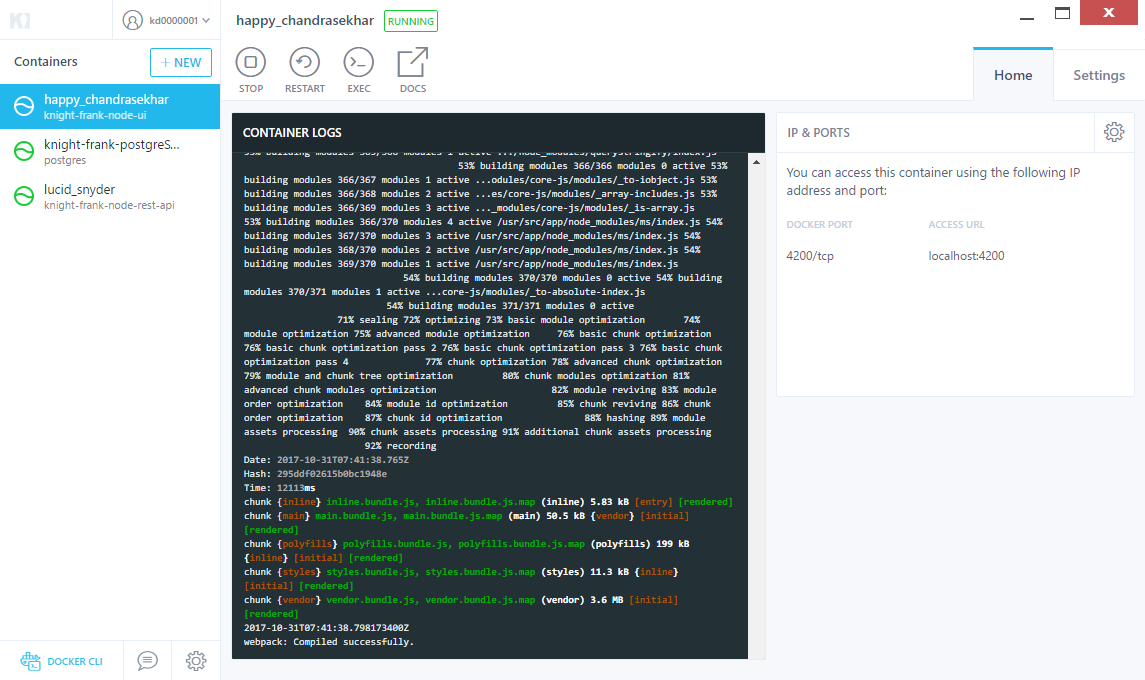
Приложение, опубликованное в облаке (<digitalocean.com>), можно увидеть по ссылке: <http://138.68.185.190>

Решение размещено по трём Docker-контейнерам Linux, содержащим уровни приложения.

Для развёртывания приложения на локальной машине достаточно запустить \_docker-compose.cmd (при условии, что на локальной машине установлен Docker). По команде будут запущены контейнеры:

* \Database\Scripts\Dockerfile - сервер базы данных (PostgreSQL)
* \Service\Rest\Dockerfile - веб-сервис (Node.js)
* \UI\ Dockerfile - интерфейс пользователя (Angular 4, на веб-сервере Ngnix)

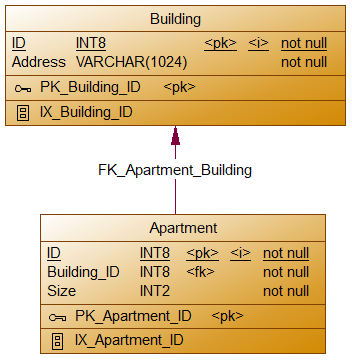
Используя Kitematic, можно наблюдать на локальной машине три запущенных контейнера:



# База данных

В прилагаемом архиве папка \Database

Согласно требованию, используется PostgreSQL, привожу диаграмму физической модели (Sybase PowerDesigner):



В прилагаемом архиве (папка “Database”) имеется диаграмма и скрипты pdplSQL создания структуры таблиц, а также генерации тестовых данных, со случайным их распределением.

# 

# Серверная часть

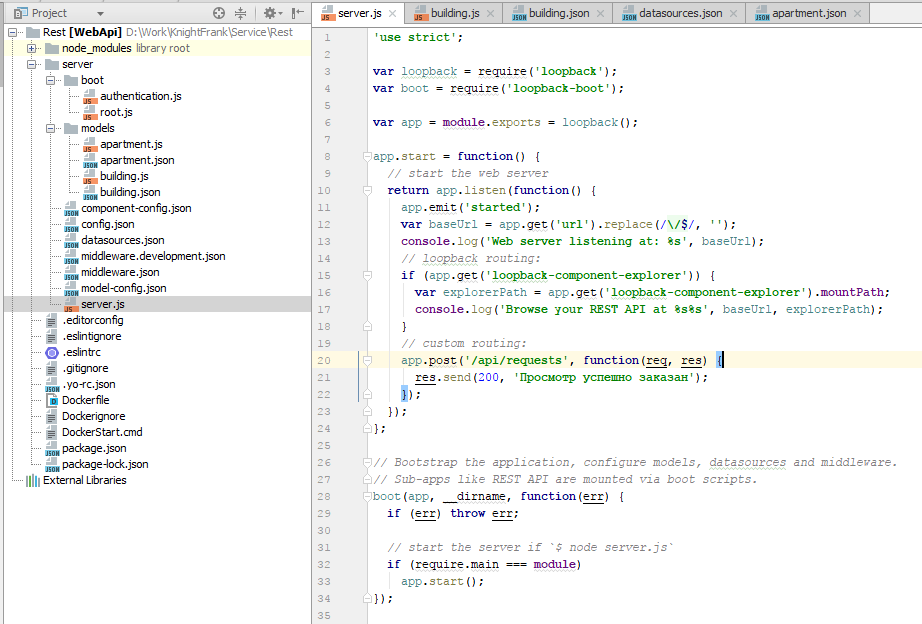
В прилагаемом архиве папка \Service\Rest

Согласно требованию, используется Node.js в качестве платформы для REST-сервиса. Используются библиотеки Express.js и основанная на ней Loopback.io, добавляющая возможности ORM-системы для реляционных данных, а также возможность генерации VIEW-модели.

Loopback позволяет как читать данные из реляционных таблиц, так и выполнять хранимые функции и SQL-запросы с параметрами, при необходимости выборок на стороне сервера.

Для данного решения это оказалось необходимым, т.к. для выборок понадобилась агрегация и соединение таблиц, что оптимально делать в реляционной СУБД, а не на стороне сервиса.

Структура серверной части проекта:

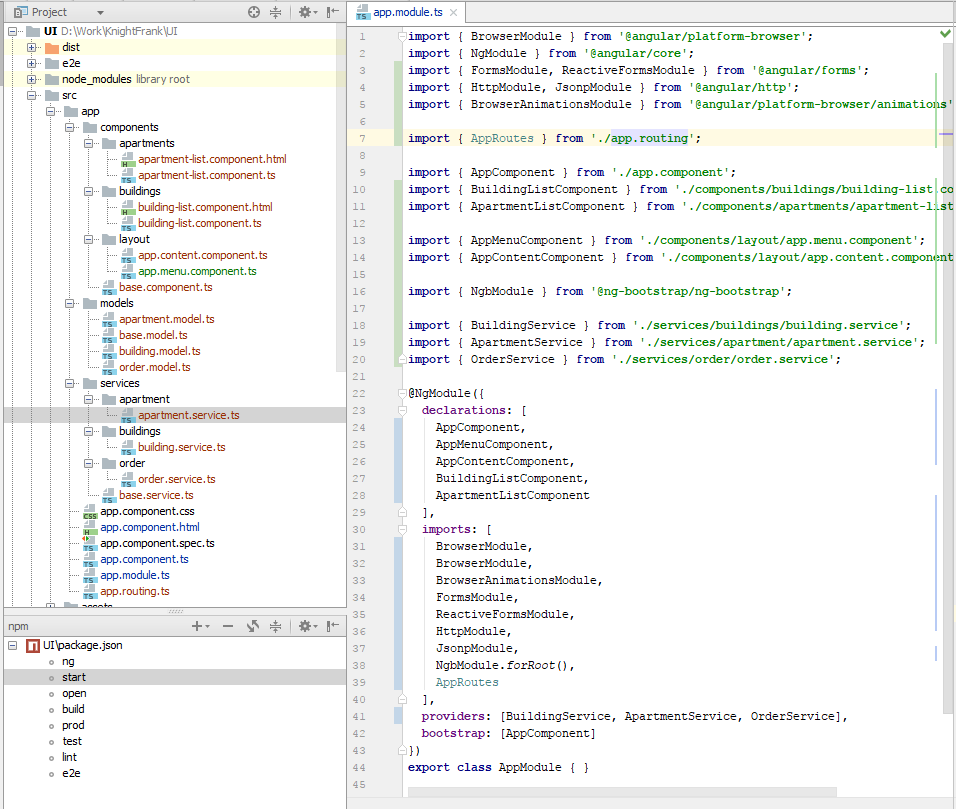


# 

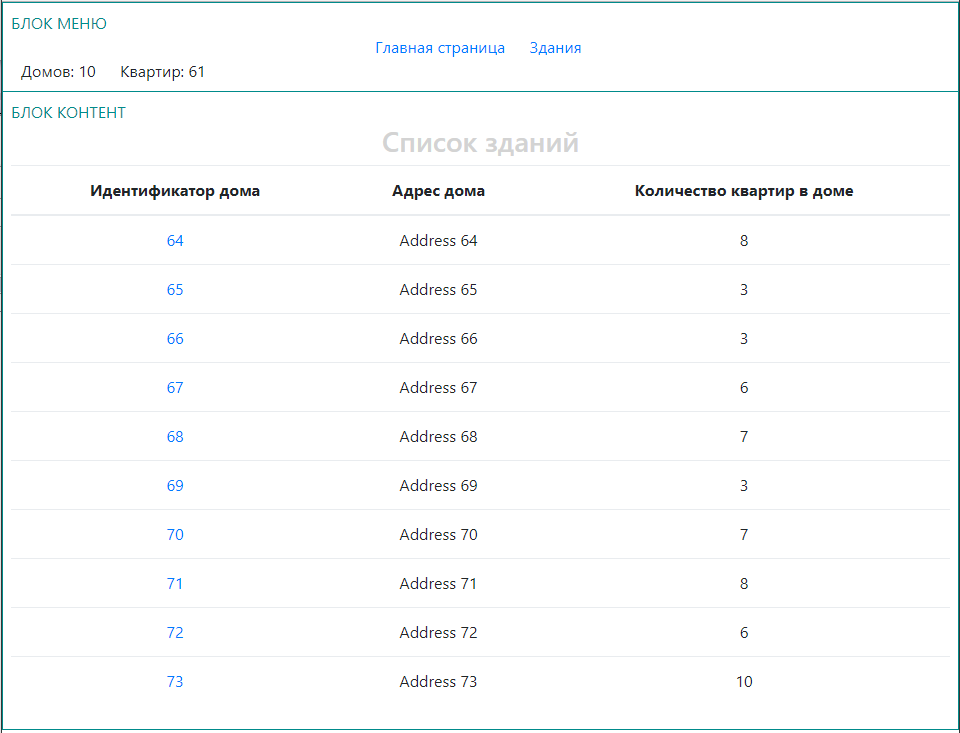
# Клиентская часть

В прилагаемом архиве папка \UI

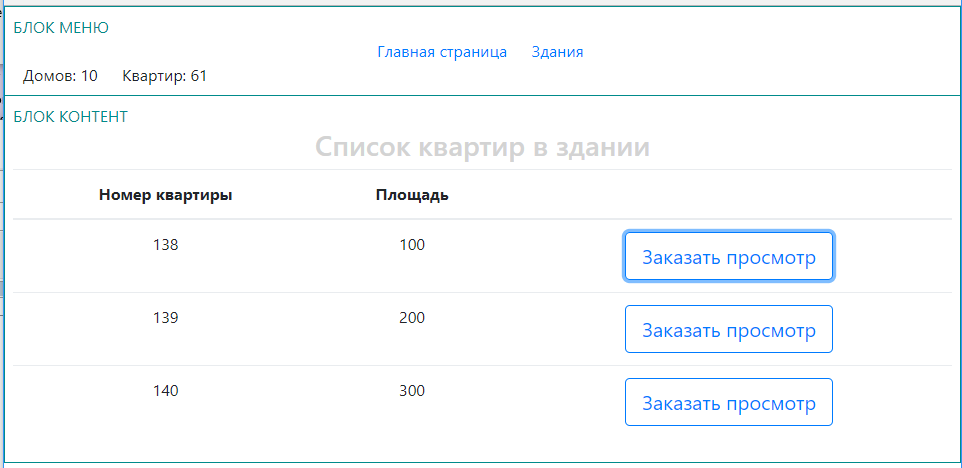
Структура клиентской части проекта:



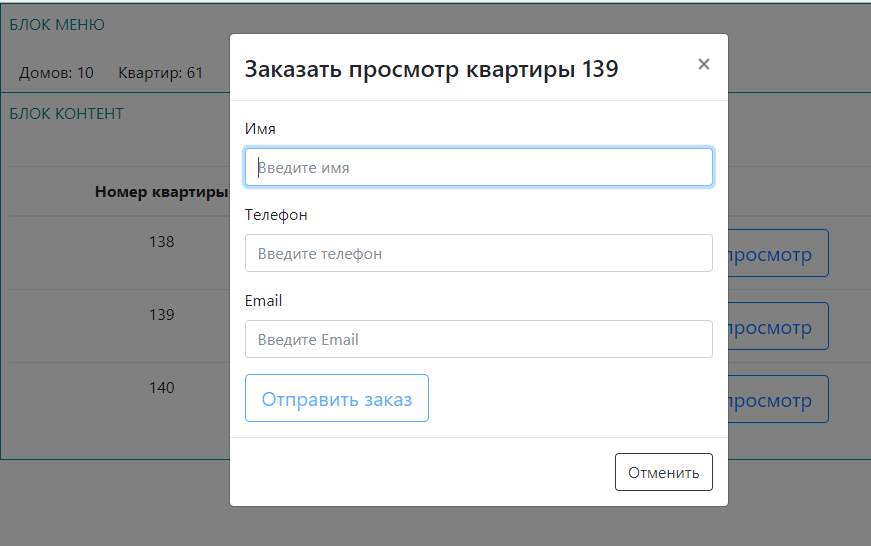
Ниже скриншот формы “Список зданий, в html-разметке специально показаны границы областей – “БЛОК МЕНЮ” и “БЛОК КОНТЕНТ” (в контенте располагается router-outlet, в который подгружаются все прочие компоненты, вызываемые из меню):



Ниже скриншот формы со списком квартир, вызываемой по клику на идентификаторе дома:



По клику на кнопке “Заказать просмотр” открывается модальное окно ng-bootstrap с формой, позволяющей оправить привязанную к форме модель Order методом Post на Rest-сервис.



В случае успешной отправки, (после отработки Promise), на форме будет показано сообщение об этом.

# 

# Тестирование