

# Описание решения

## Содержание

Общее описание.....	1
База данных .....	2
Серверная часть.....	3
Клиентская часть .....	4
Тестирование .....	6

## Общее описание

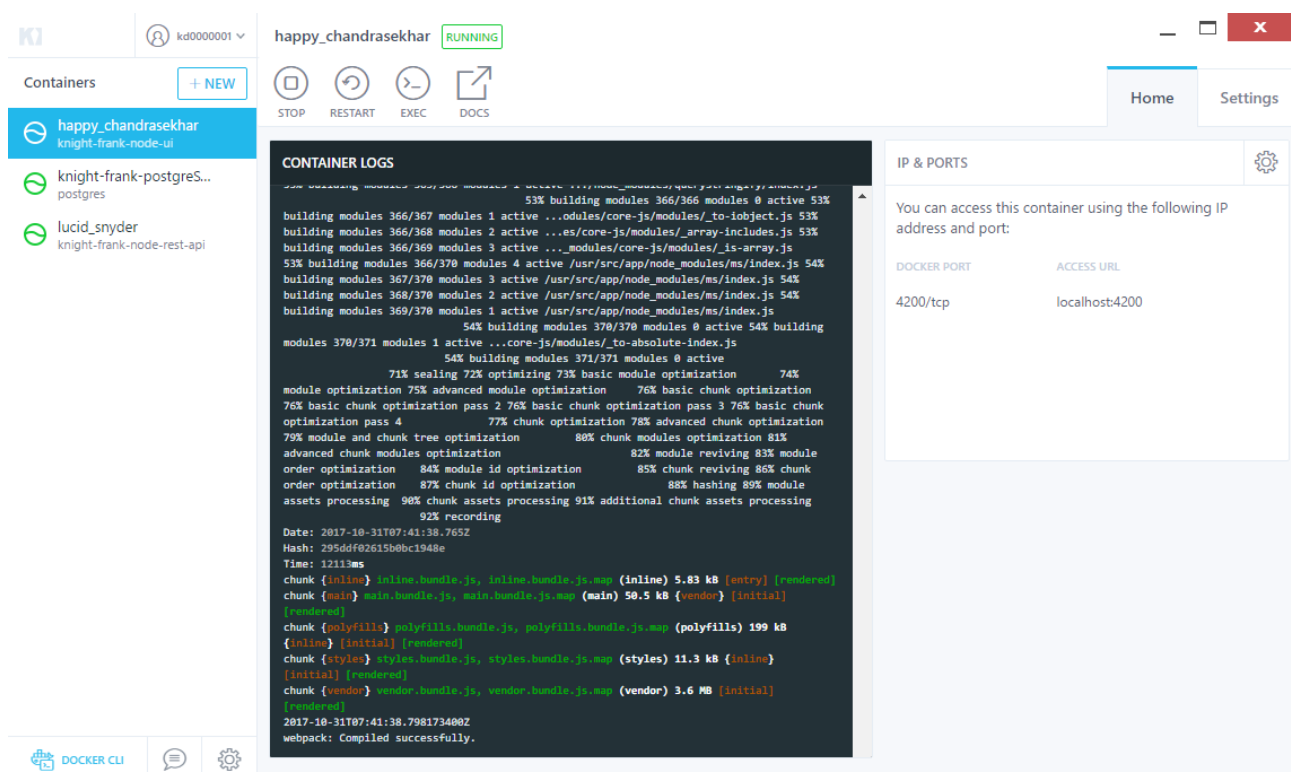
Приложение, опубликованное в облаке (digitalocean.com), можно увидеть по ссылке: <http://138.68.185.190>

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 Nginx)

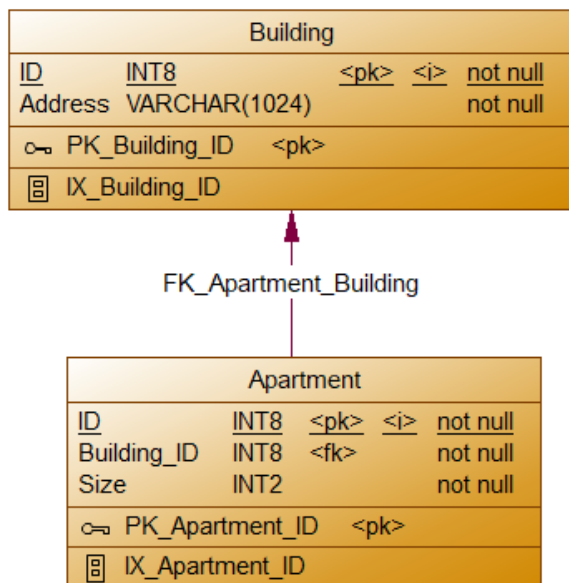
Используя Kitematic, можно наблюдать на локальной машине три запущенных контейнера:



## 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

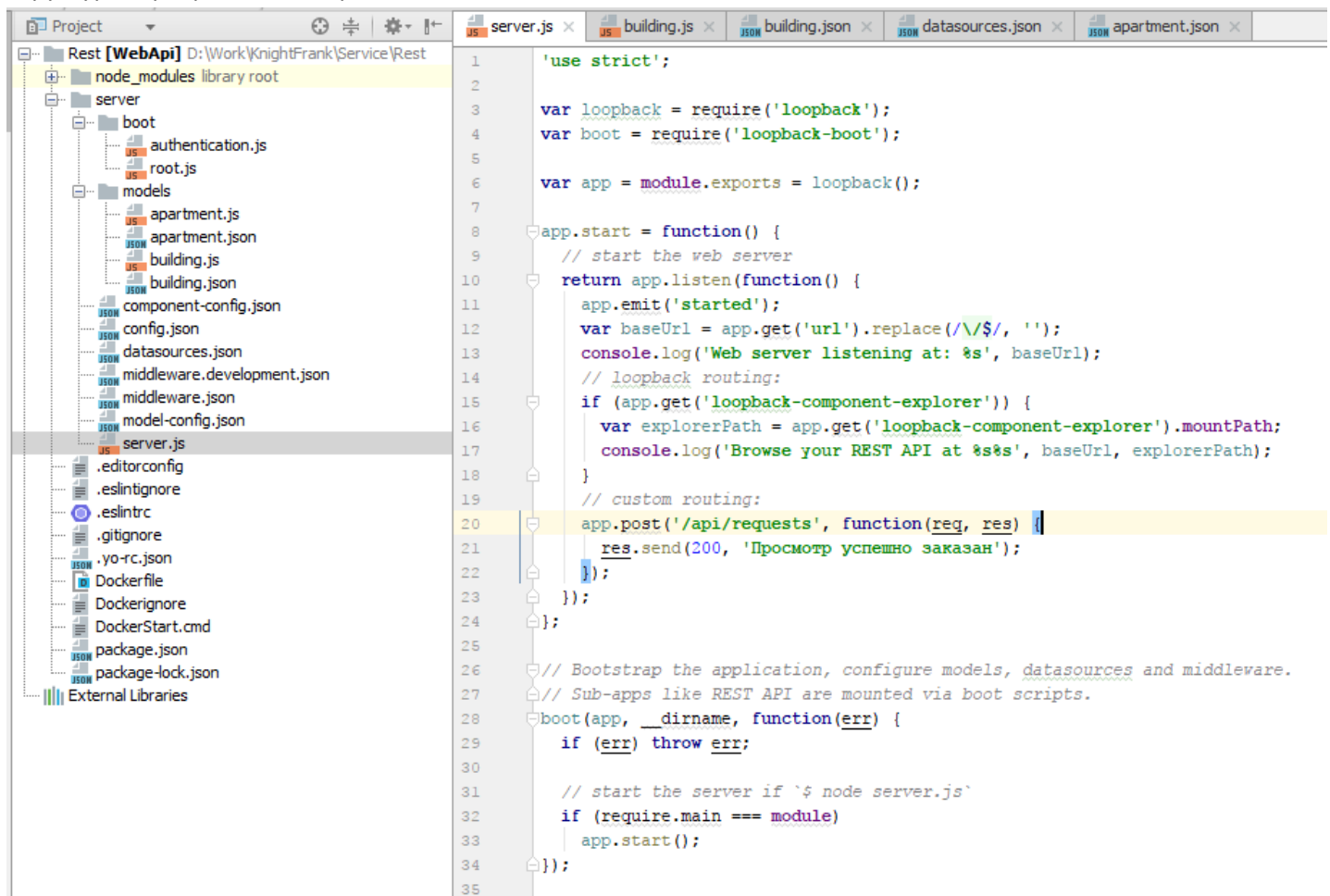
В прилагаемом архиве папка \Service\Rest

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 выборки 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:



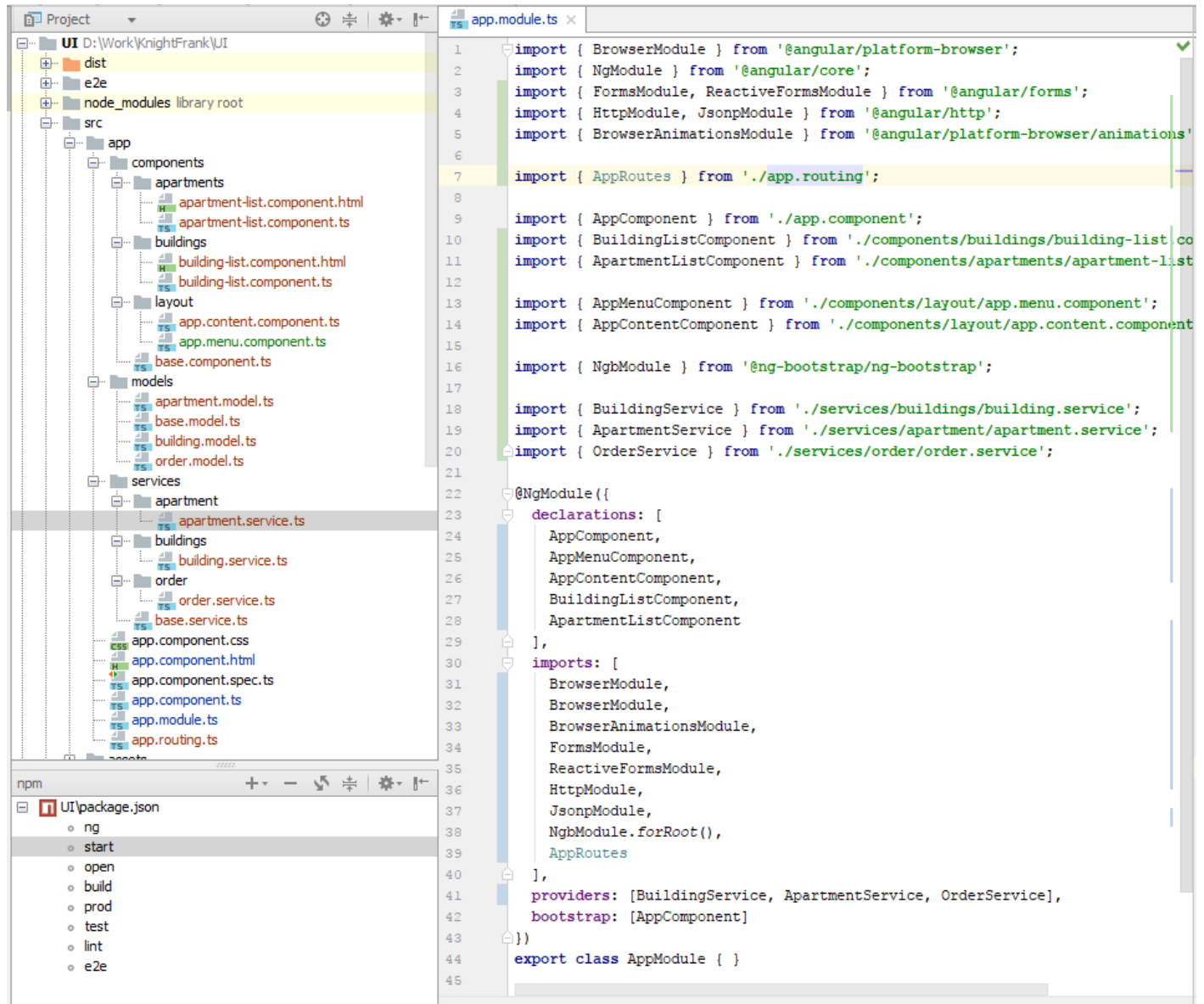
The screenshot displays an IDE with a project structure on the left and the `server.js` file open in the editor. The project structure shows a `Rest [WebApi]` directory containing `node_modules`, `server`, `boot`, `models`, `component-config.json`, `config.json`, `datasources.json`, `middleware.development.json`, `middleware.json`, `model-config.json`, `server.js`, `.editorconfig`, `.eslintignore`, `.eslintrc`, `.gitignore`, `.yo-rc.json`, `Dockerfile`, `Dockerignore`, `DockerStart.cmd`, `package.json`, and `package-lock.json`. The `server.js` file contains the following code:

```
1 'use strict';
2
3 var loopback = require('loopback');
4 var boot = require('loopback-boot');
5
6 var app = module.exports = loopback();
7
8 app.start = function() {
9   // start the web server
10  return app.listen(function() {
11    app.emit('started');
12    var baseUrl = app.get('url').replace(/\/$/, '');
13    console.log('Web server listening at: %s', baseUrl);
14    // loopback routing:
15    if (app.get('loopback-component-explorer')) {
16      var explorerPath = app.get('loopback-component-explorer').mountPath;
17      console.log('Browse your REST API at %s%s', baseUrl, explorerPath);
18    }
19    // custom routing:
20    app.post('/api/requests', function(req, res) {
21      res.send(200, 'Просмотр успешно заказан');
22    });
23  });
24 };
25
26 // Bootstrap the application, configure models, datasources and middleware.
27 // Sub-apps like REST API are mounted via boot scripts.
28 boot(app, __dirname, function(err) {
29   if (err) throw err;
30
31   // start the server if '$ node server.js'
32   if (require.main === module)
33     app.start();
34 });
35
```

## 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):

БЛОК МЕНЮ		
<a href="#">Главная страница</a> <a href="#">Здания</a>		
Домов: 10   Квартир: 61		
БЛОК КОНТЕНТ		
Список зданий		
Идентификатор дома	Адрес дома	Количество квартир в доме
64	Address 64	8
65	Address 65	3
66	Address 66	3
67	Address 67	6
68	Address 68	7
69	Address 69	3
70	Address 70	7
71	Address 71	8
72	Address 72	6
73	Address 73	10

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

БЛОК МЕНЮ		
<a href="#">Главная страница</a> <a href="#">Здания</a>		
Домов: 10   Квартир: 61		
БЛОК КОНТЕНТ		
Список квартир в здании		
Номер квартиры	Площадь	
138	100	Заказать просмотр
139	200	Заказать просмотр
140	300	Заказать просмотр

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

БЛОК МЕНЮ

Домов: 10    Квартир: 61

БЛОК КОНТЕНТ

Номер квартиры

138

139

140

Заказать просмотр квартиры 139

Имя

Введите имя

Телефон

Введите телефон

Email

Введите Email

Отправить заказ

Отменить

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

БЛОК МЕНЮ	Главная страница   Здания	
Домов: 10    Квартир: 61		
БЛОК КОНТЕНТ	Список квартир в здании	
Номер квартиры	Площадь	
138	100	Заказать просмотр
139	200	Заказать просмотр
140	300	Заказать просмотр
Заказ на квартиру 139 успешно отправлен		

## Тестирование