# Задание на выпускную квалификационную работу

# Реферат

Содержание

[Задание на выпускную квалификационную работу 1](#_Toc484063129)

[Реферат 1](#_Toc484063130)

[Введение 3](#_Toc484063131)

[Обзор аналогичных разработок и постановка задачи 4](#_Toc484063132)

[Аналитический обзор литературы 4](#_Toc484063133)

[Постановка задачи 4](#_Toc484063134)

[Общее описание 4](#_Toc484063135)

[Страницы сайта 5](#_Toc484063136)

[Основные структурные части сервиса и их функции 6](#_Toc484063137)

[Описание архитектуры 10](#_Toc484063138)

[Одностраничное приложение (SPA) 10](#_Toc484063139)

[Система авторизации 12](#_Toc484063140)

[Архитектура сервера 13](#_Toc484063141)

[Описание разработки серверной части 15](#_Toc484063142)

[База данных 15](#_Toc484063143)

[API 17](#_Toc484063144)

[Общее описание 17](#_Toc484063145)

[Описание методов API 19](#_Toc484063146)

[Описание разработки клиентской части 24](#_Toc484063147)

[Используемые возможности Angular 24](#_Toc484063148)

[Маршрутизация 24](#_Toc484063149)

[Сервисы 25](#_Toc484063150)

[Сервис для работы с пользователями 25](#_Toc484063151)

[Сервис для работы с информацией о кафе 27](#_Toc484063152)

[Сервис для работы с заказами и столиками 28](#_Toc484063153)

[Директивы 29](#_Toc484063154)

[Директива для отображения оценок 29](#_Toc484063155)

[Директива для управления авторизацией 29](#_Toc484063156)

[Директива конструктора схемы помещения кафе 29](#_Toc484063157)

[Страницы и их контроллеры 33](#_Toc484063158)

[Главная страница 33](#_Toc484063159)

[Страница кафе 33](#_Toc484063160)

[Страница регистрации 34](#_Toc484063161)

[Страница авторизации 34](#_Toc484063162)

[Личный кабинет пользователя 35](#_Toc484063163)

[Панель администратора кафе 35](#_Toc484063164)

[404 37](#_Toc484063165)

[Модальные окна и их контроллеры 37](#_Toc484063166)

[Окно заказа столика 37](#_Toc484063167)

[Окно обновления изображения 37](#_Toc484063168)

[Окно для написания отзыва о кафе 37](#_Toc484063169)

[Окно подтверждения удаления пользователя 37](#_Toc484063170)

[Заключение 38](#_Toc484063171)

[Список литературы 39](#_Toc484063172)

# Введение

На сегодняшний день сфера обслуживания, в том числе услуг общественного питания, в России находятся в состоянии развития. Отрываются новые заведения: кафе, бары, рестораны и т.д. Это рождает конкуренцию между ними, а следовательно, заставляет повышать качество предоставляемых услуг, искать новые методы привлечения клиентов. С другой стороны потенциальные клиенты могут быть дезориентированы многообразием выбора. Предпринимателям нужны инструменты для привлечения клиентов, клиенты заинтересованы в удобном сервисе для выбора места отдыха. Разрабатываемый web-сервис может помочь и тем и другим.

Размещение информации о заведении, в том числе ссылок на его сайт или страницу в соц. сети, положительно влияет на позиции кафе в поисковых системах. А значит, сервис помогает увеличивать количество потенциальных клиентов, знающих о существовании заведения.

Функционал просмотра плана помещения кафе и выбора наиболее подходящего потребностям клиента столика повышает качество обслуживания клиентов. Кроме того, это подталкивает посетителей к посещению заведения: чтобы быть уверенными, что им достанется наиболее подходящий под их вкусы и потребности столик, они его закажут и придут именно в это заведение.

Web-сервис помогает клиентам в выборе заведения, так как в нем есть система оценок кафе, а также механизм сортировки и фильтрации кафе по разным критериям. По оценкам и комментариям других пользователей можно понять реальные плюсы и минусы кафе. Механизм сортировки и фильтрации помогает сделать выборку интересующих посетителя категорий кафе, тем самым упрощая процесс выбора.

# Обзор аналогичных разработок и постановка задачи

## Аналитический обзор литературы

## Постановка задачи

### Общее описание

Цель работы: разработать web-сервис заказа столиков в кафе. Web-сервис представляет собой распределённое клиент-серверное приложение. Клиент должен быть выполнен в виде одностраничного приложения. Сервер должен предоставлять API для предоставления данных клиенту.

Назначение web-сервиса:

* Агрегация информации о разных кафе, в том числе информации о мнении посетителей
* Предоставление возможности удобного бронирования за собой мест в кафе на определённое время для посетителей
* Предоставление возможности получения обратной связи от клиентов для администраторов кафе

Сайт должен предоставлять пользователям доступ к следующей информации:

* Общая информация о кафе (название, адрес, контакты, способы оплаты, кухня, средний чек)
* Информацию о количестве посетителей кафе относительно друг друга (не конкретные цифры) и средняя оценка посетителей по 5-бальной шкале
* Отзывы зарегистрированных пользователей о кафе (оценка по 5-бальной шкале и комментарий)
* Информацию о доступности любого столика в кафе в любой момент времени

При входе в web-приложение пользователь имеет возможность авторизоваться, зарегистрироваться или остаться анонимным пользователем, которому доступен только просмотр информации о заведениях. При регистрации доступны две роли: роль простого пользователя и роль мини-администратора, представителя ресторана. У той и другой роли есть собственные личные страницы: панель мини-администратора и личный кабинет пользователя. В личном кабинете пользователь может управлять информацией о себе и своими заказами столиков. Авторизованному пользователю доступна возможность заказывать столики в ресторанах и оценить их. В панели мини-администратора есть возможность управлять информацией о ресторане, в том числе схемой помещения.

На главной странице приложения доступен список ресторанов с возможностью фильтрации по перечисленным в схеме критериям (рейтинг, популярность и т.д.). Со страницы списка можно перейти на страницу конкретного ресторана, где доступна вся информация о нём и возможность зарезервировать столик авторизованным пользователям.

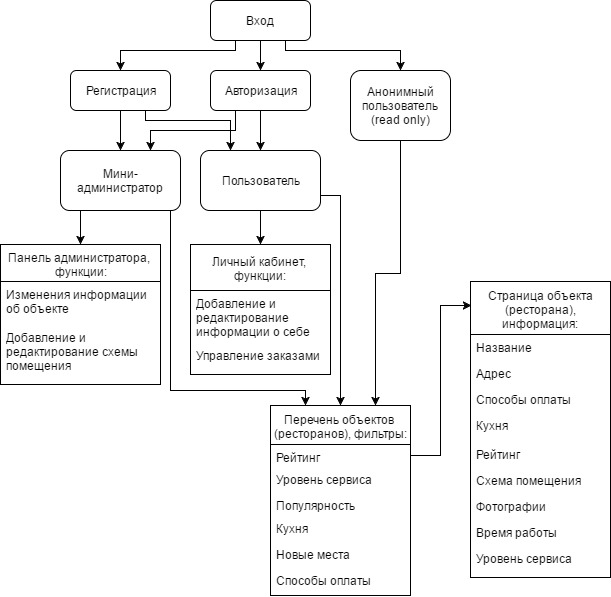


Рисунок 2.1 - Блок-схема web-приложения сервиса заказа столиков в ресторане

### Страницы сайта

Страницы должны быть все связанны между собой (Рисунок 2.2), чтобы по ссылкам можно было попасть на любую страницу сайта, кроме специальной страницы 404. Список страниц:

* Главная станица со списком кафе
* Страница одного кафе
* Личный кабинет пользователя (доступен только авторизованным пользователям)
* Панель администратора (доступна только авторизованным администраторам кафе)
* Страница регистрации
* Страница авторизации
* Страница 404 (если запрашиваемой страницы нет на сайте)

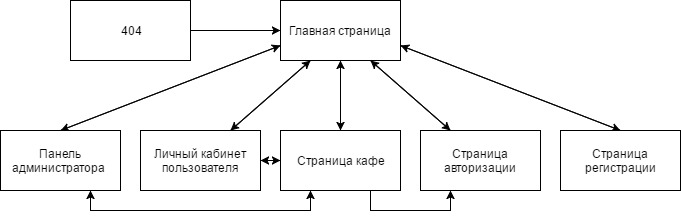


Рисунок 2.2 – Навигация по сайту, система ссылок

### Основные структурные части сервиса и их функции

#### Система авторизации

В сервисе существуют три роли: анонимного пользователя, посетителя кафе и администратора кафе. У каждого у них есть функции, которые они имеют право совершать и соответствующие им методы API.

Анонимный пользователь может смотреть информацию о кафе, ему доступны функции подбора кафе по интересующим параметрам и регистрации в web-приложении.

Посетитель кафе – авторизованный пользователь, и он имеет больше возможностей, чем анонимный посетитель приложения. Посетителю кафе доступны возможности заказывать столики, оставлять и удалять отзывы о кафе. Кроме того, в личном кабинете можно просматривать и удалять заказами и отзывами, изменять информацию о себе.

Администратор кафе может создать кафе, добавить информацию о нём, схему помещения, фотографию кафе. В панели администратора так же можно просматривать заказы.

#### Список кафе

В списке кафе на главной странице (Рисунок 2.9) должна выводиться краткая информация обо всех кафе: название, адрес, средний чек, рейтинг, кухня и способы оплаты. Кроме того, списком можно манипулировать с помощью следующих фильтров:

* Фильтр по вводимым словам (буквам), выбор кафе в случае совпадения значения фильтра с какой-либо информацией о кафе
* Фильтр по определённому способу оплаты
* Фильтр по определённой кухне
* Сортировка списка по следующим критериям:
  + популярности
  + рейтингу (средняя оценка по отзывам)
  + среднему чеку
  + дате создания

#### Список отзывов

На странице кафе отображается список отзывов о кафе (Рисунок 2.10) от зарегистрированных пользователей. Каждый отзыв содержит следующие поля: имя автора, дата создания, комментарий к отзыву и оценка. Кроме того, авторизованному автору отзыва доступна кнопка для удаления своего отзыва. Список отзывов сортирован по дате создания: от старых к новым.

#### Добавление отзыва о кафе

Форма добавления отзыва находится в модальном окне на странице кафе и появляется после нажатия кнопки «Добавить отзыв». Форма содержит поля формы оценка и комментарий, оба поля обязательны для заполнения. После отправки отзыва, окно закрывается, а в списке отзывов появляется новый отзыв. Кнопка добавления отзыва видна только авторизованным пользователям.

#### Заказ столика

Форма заказа находится в модальном окне на странице кафе. В ней должна быть отображена схема помещения кафе с отмеченными на ней столиками. Кроме того, в форме есть инструмент для выбора даты и промежутка времени, на которое планируется заказать столик. В зависимости от выбора времени на схеме отмечаются доступные для заказа столики. Доступность рассчитывается исходя из расписания работы кафе в выбранный день и уже совершенных ранее заказов. Выбор даты, времени и столика обязательны для отправки формы. Кнопка заказа видна только авторизованным пользователям.

#### Создание кафе и обновления информации о кафе

Для этих двух функций используется одна и так же форма на странице панели администратора. Отличие состоит в том, что при отсутствии созданного кафе на панели отображается уведомление и форма пуста, а в случае, если кафе создано, в полях формы вставлена актуальная информация. Поля формы:

* Название
* Адрес
* Средний чек
* Кухня
* Способы оплаты (возможно несколько значений) (Рисунок 2.13)
* Контакты. Есть возможность добавлять произвольное количество контактов. Существуют контакты в виде текста (для номера телефона, адреса офиса и т.п.) и в виде ссылки (для ссылок на соц. сети или сайты). Для добавления контакта должно быть заполнено 3 поля: тип контакта, название и значение (ссылка на ресурс). После добавления контакта он отображается в списке в элементе формы. От туда его можно удалить с помощью кнопки с изображением крестика с левой стороны элемента списка.
* Время работы кафе. Для каждого дня недели (для одного или сразу для нескольких) можно задать время открытия и закрытия кафе. Пропущенные дни недели считаются нерабочими. Как и на элементе добавления контактов, после добавления расписания на день, оно отображается в списке в элементе формы и его можно удалить. При добавлении сразу нескольких дней недели каждый день добавляется в список отдельно.

#### Конструктор схемы помещения кафе

На странице административной панели должна быть возможность редактировать схему кафе и расположение столиков на ней. Для этого нужен конструктор. Конструктор должен иметь следующий функционал:

* Изменение изображения схемы помещения
* Создание «образов» столиков (фигур, символизирующих столик)
* Изменение расположения (координат) столиков на схеме с помощью перетаскивания мышью
* Изменение размеров столиков
* Удаление столиков
* Добавление информации о том, сколько мест доступно за столиком
* Группировка столиков с помощью выделения области, со столиками, попавшими в область, можно совершить операции изменения положения и удаления одновременно
* Очистка схемы

При клике на кнопку «Установить изображение» появляется модальное окно с формой выбора файла. После отправки формы модальное окно закрывается, а выбранное изображение появляется в конструкторе.

#### Список заказов

Список заказов существует в двух вариантах: список заказов для конкретного кафе, доступный в панели администратора и список заказов от конкретного пользователя, доступный в личном кабинете. Список включает в себя следующие поля (поля названия кафе или имени пользователя могут быть опущены там, где это и так понятно):

* Название кафе
* Номер столика
* Количество мест за столиком
* Время заказа
* На кого сделан заказ

# Описание архитектуры

## Одностраничное приложение (SPA)

SPA – это приложение, выполняющие все внутри браузера и никогда не производящее полной перезагрузки страницы. Это значит, что вся логика, обработка данных, пользовательский поток выполнения и отправка шаблонов могут быть организованы в браузере. Классическим примером SPA является Gmail. На странице, помимо разнообразных наборов данных, показываются различные представления, но сама страница никогда не перезагружается. Для создания SPA будет использоваться JavaScript-фреймворк Angular.

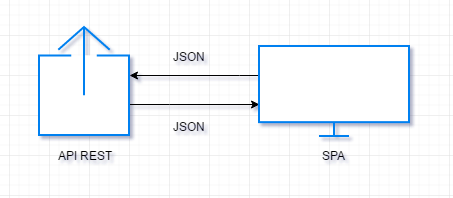


Рисунок 4.2.2 – Одностраничное приложение взаимодействует с REST API, поставляющему данных в формате JSON

Для того чтобы SPA взаимодействовало с сервером, создаётся API (Рисунок 4). Это архитектурный стиль взаимодействия компонентов распределённого приложения в сети. В сети Интернет вызов удалённой процедуры может представлять собой обычный HTTP-запрос (обычно «GET» или «POST»), а необходимые данные передаются в качестве параметров запроса. Для разработки API используется фреймворк Express.

Кроме данных, с сервера загружаются шаблоны – HTML-файлы, куда с помощью специальных директив вставляются данные из модели. Этот процесс называется компиляцией. При изменении модели моментально изменяется представление (Рисунок ).

Angular – упрямый фреймворк, это значит, что он жестко задаёт архитектуру приложения, а именно использование паттерна проектирования MVC (Рисунок ). Между моделью (представлением данных и внутреннего поведения) и внешним представлением располагается контроллер, связывающий события и отображения в представлении с методами модели. Для формирования модели используются вышеупомянутые HTTP-запросы на сервер, представление – это шаблоны, а в контроллеры соединяют их с помощью логики приложения, обработки событий, совершаемых пользователем web-сервиса.

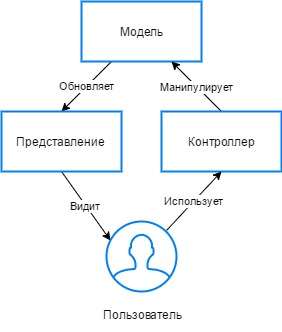
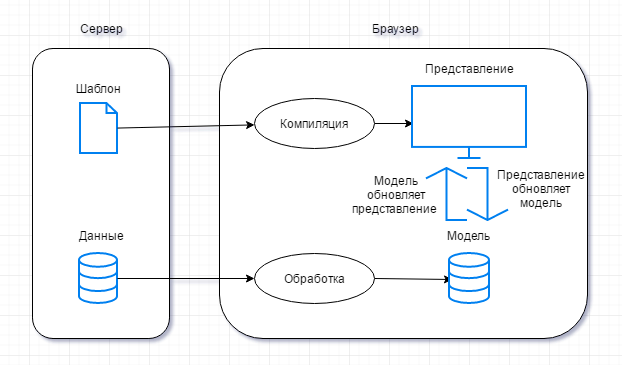


Рисунок Паттерн проектирования MVC

Рисунок 4.2.1 – Двусторонняя привязка данных – представление и модель обрабатываются в браузере и связываются, причём каждый немедленно обновляет другого

Angular позволяет создавать разные сущности, помогающие структурировать приложение. Это способствует разделению приложения на отдельные независимые логические части, повторному использованию кода, созданию отдельных областей видимости для разных частей приложения. Например, директивы в Angular позволяют вынести часть HTML-кода и относящейся к нему логики в собственный HTML тег или атрибут.

## Система авторизации

В сервисе существуют две роли: простого пользователя (посетителя кафе) и администратора кафе. У каждого у них есть функции, которые они имеют право совершать и соответствующие им методы API.

Для аутентификации пользователя в системе используется уникальный маркер – JSON Web Token (JWT). Для того, чтобы сгенерировать его, требуется хеш, email и тип пользователя. Хеш – это зашифрованные значение объединённых пароля, созданного пользователем, и соли, случайной строки, генерируемой приложением. В базе данных пароль храниться также в виде хеша и соли. При таком подходе все маркеры уникальны, а пароли хорошо защищены.

Токен актуален в течение 7 дней, после чего требуется повторная авторизация пользователя. Между запросами к API токен должен хранится в локальном хранилище браузера. В каждый запрос, требующий подтверждения прав пользователя, нужно включать заголовок со значением токена.

При входе пользователь отправляет свои учетные данные на сервер, сервер их проверяет и возвращает идентификационный маркер браузеру. Браузер сохраняет его для дальнейшего повторного использования (Рисунок 2.3).

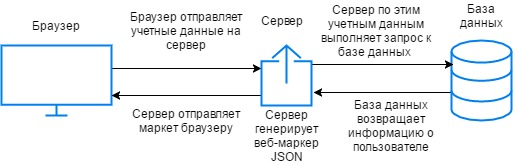


Рисунок 2.3 Поток выполнения входа в приложение

Во время действия сеанса пользователя он должен иметь возможность сменить представление, а браузер должен знать, что может видеть пользователь. Для этого он расшифровывает часть маркера и в соответствии с этой информацией меняет представление (Рисунок 2.4).

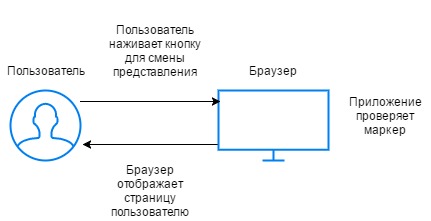


Рисунок 2.4 Приложение проверяет, что следует показать пользователю, в зависимости от его роли

Для того, чтобы делать некоторые запросы в API нужна аутентификация. Поэтому в заголовке запроса к этим методам должен быть маркер JWT (Рисунок 2.5).

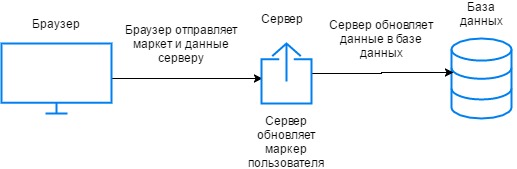


Рисунок 2.5 Приложение проверяет, что следует показать пользователю, в зависимости от его роли

## Архитектура сервера

На рисунке изображены структура серверной части web-сервиса. Главная задача сервера – проверить, имеет ли право клиент делать запрос к данной точке API, выбрать соответствующие запросу данные из БД, «упаковать» в JSON и вернуть клиенту. Если на каком-то этапе произошла ошибка – вернуть клиенту сообщение об ошибке.

Когда на сервер поступает запрос, нужно в первую очередь определить, какой у него тип (GET, POST и т.д.) и к какому URL-пути он относиться. За это отвечает блок «Маршруты». В этом блоке указано, какой метод обрабатывает тот или иной запрос, а также, нужно ли быть авторизованным для доступа к этому методу API. На этом этапе не определяется, действительно ли пользователь авторизован, а только задается требование об авторизации.

Проверка авторизации происходит в блоке Passport. Он так называется, поскольку для этого используется одноименная js-библиотека. Она помогает реализовать описанный в прошлом разделе механизм авторизации с помощью токенов. Passport делает запрос к базе данных для поиска пользователя с email, указываемом для авторизации. Затем он поверяет, подходит ли пароль. В случае неудачи, возвращает сообщение об ошибке ("Неверное имя" или "Неверный пароль"). В случае успеха добавляет в тело запроса поле «payload» с данными о пользователе – имя, email, тип. Тело запроса позже попадёт в соответствующий запросу метод в блоке «Логика».

Как понятно из названия блок «Логика» отвечает за логику API, он проверяет тип авторизованного пользователя, делает запросы к базе данных, возвращает правильно структурированные JSON-объекты клиенту. Каждому маршруту соответствует свой метод в этом блоке, для удобства они разделены на несколько контроллеров:

* Работа с пользователями (authentication.js)
* Информация о кафе (café.js)
* Отзывы о кафе (reviews.js)
* Заказы и работа со семой помещения (order.js)

Блоки Pathport и Логика обращаются не напрямую к базе данных MongoDB, а к Mongoose.

Mongoose – это ORM для работы с MongoDB в Node.js. ORM предназначены для того, чтобы обращаться не к таблицам базы данных, а к объектной модели таблиц. Это больше соответствует объектно-ориентированной парадигме программирования, в коде удобнее оперировать объектами, их полями и методами, чем строками запросов к базе данных. Кроме того, эти Mongoose берёт на себя настройку взаимодействия приложения и базы данных. К модели таблиц можно добавлять свои методы.

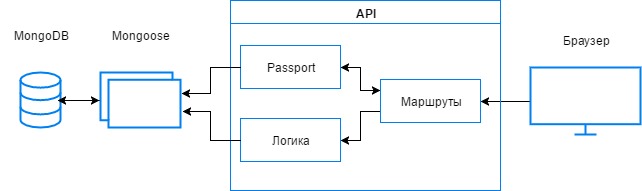


Рисунок Архитектура сервера

# Описание разработки серверной части

## База данных

База данных состоит из 7 таблиц (Рисунок):

* Cafes (информация о кафе)
* timeSchema (расписание работы кафе)
* contactSchema (контакты кафе)
* reviewsSchema (информация об отзывах)
* Users (общая информация о пользователях)
* Clients (информация о посетителях кафе)
* Admins (информация о администраторах кафе)
* Tables (информация о столиках)
* Orders (информация о заказах)

В таблицах timeSchema, contactSchema и reviewsSchema не указано id кафе, так как таблица Cafes является составной, внутри неё хранятся объекты с отзывами, контактами и расписанием работы кафе. Эти данные используются только в связке с кафе, их имеет смысл хранить вместе.

Таблица timeSchema используется для хранения расписания работы кафе. В ней содержится следующая информация: номер дня недели, имя дня недели, время открытия, время закрытия, открыто ли кафе в этот день.

В таблице contactSchema хранится информация о контактах: название, значение контакта и флаг, является ли контакт ссылкой. Это сделано для того, чтобы оформлять ссылки должным образом.

В таблице reviewsSchema хранится отзывы на кафе: автор отзыва, дата создания отзыва, комментарий и оценка. Оценка может быть от 0 до 5, это минимальное и максимальное значения поля.

Таблица Cafes содержит следующую информацию о кафе: название, адрес, кухня, способ оплаты (массив значений типа String), расписание кафе (массив значений типа timeSchema длинной 7, по числу дней в неделе), контакты (массив значений типа contactSchema), отзывы (массив значений типа reviewsSchema), рейтинг кафе (значение от 0 до 5), дата создания, средний чек, количество заказов (они отражают популярность кафе), изображение кафе, изображение схемы помещения кафе.

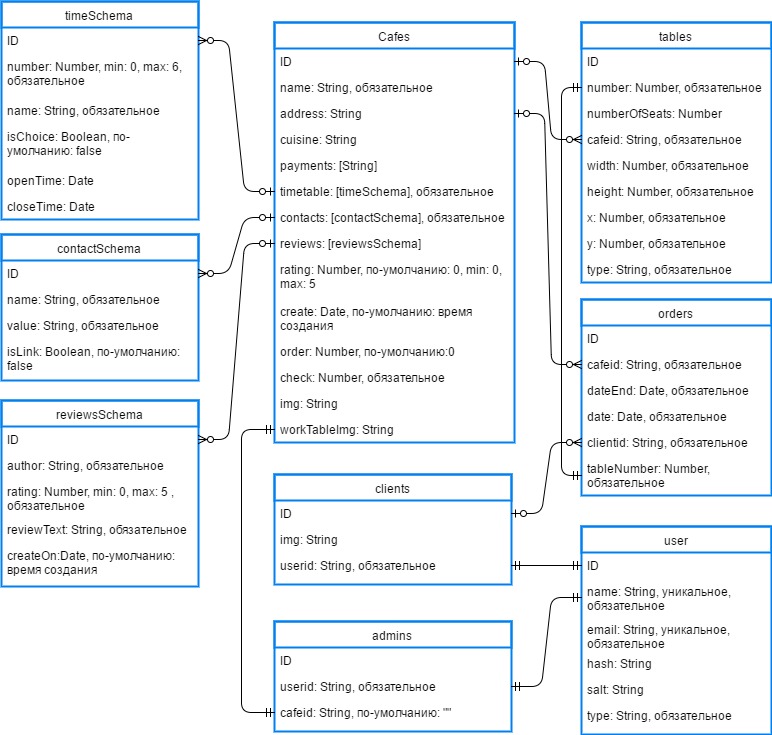


Рисунок Схема базы данных

Таблица User предназначена для хранения данных общих, для всех пользователей любого типа. Это поля с именем пользователя (должно быть уникальным), email пользователя (должен быть уникальным), тип пользователя («admin» или «client»). Вместо того, чтобы хранить пароль пользователя в явном виде, в таблице хранятся значения соли и хеша.

Для разных типов пользователей используются разные таблицы: clients для посетителей и admins для администраторов кафе. Они связаны с таблицей User отношением один к одному. В них обоих хранится id записи в таблице User. Кроме этого в admins хранится id записи в таблице café, соответствующая созданному этим администратором кафе. В Clients есть поле для хранения изображения.

В Tables содержится список столиков, таблица связана с таблицой Cafes отношением многие к одному (много столиков в одном кафе). Включает следующие поля: номер столика, количество мест для посетителей за столиком, id кафе, в котором стоит столик. Ещё в таблице хранятся данные для отображения столиков на схеме помещения кафе: координаты (x и y), размеры (ширина и высота) и тип столика («circle» или «rectangle»).

Таблица Orders предназначена для хранения заказов, в ней хранится следующая информация: номер столика, id кафе, id клиента, время начала заказа, время конца заказа.

## API

### Общее описание

Вызов методов API нужен, чтобы получать или передать информацию серверу. Вызов представляет собой обращение по определённому адресу с использованием HTTP-метода (GET, POST и др.) и передачи определённых параметров (например, могут понадобится id кафе, отзыва, заказа и т.д.). В таблице 2.1 представлен список разработанных методов API и полная информация для их использования. Методы API возвращают в качестве результата JSON объект и код ответа сервера. В случае ошибки код 404 и объект с полем message, в котором содержится текст ошибки.

Отличительной особенностью API является отсутствие сохранения состояния. Иными словами, сервер не должен отслеживать, хранить или использовать текущую информацию о клиенте или его действиях. Независимость от состояния означает, что логика работы методов API не должна зависеть от вызовов методов API, сделанных ранее. HTTP-методы используется для разных CRUD (create, read, update, delete) операций:

* GET – чтение данных
* POST – запись данных
* PUT – обновление данных
* DELETE – удаление данных

Таблица 2.1

Описание методов API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Действие** | **Метод** | **Путь URL** | **Параметры** | **Доступ** |
| Получение списка кафе | get | /api/cafe |  | Доступен всем |
| Получение подробной информации о кофе | get | /api/cafe/cafeid | Cafeid – id кафе |
| Регистрация | post | /api/user/register |  |
| Авторизация | post | /api/user/login |  |
| Получение информации о схеме помещения кафе | get | /api/cafe/cafeid/workTable | Cafeid – id кафе |
| Получение краткой информации о заказах в кафе (без имени заказчика) | get | /api/cafe/cafeid/orders | Cafeid – id кафе |
| Получение информации о пользователе | get | /api /user/client |  | Доступен авторизованным пользователям |
| Изменение фотографии пользователя | post | /api/user/client/image |  |
| Создание отзыва о кафе | post | /api/cafe/cafeid/reviews | Cafeid – id кафе |
| Удаление отзыва | delete | /api/cafe/cafeid/reviews/reviewid | Cafeid – id кафе  Reviewid – id отзыва |
| Создание заказа | post | /api/user/client/order |  |
| Удаление заказа | delete | /api/user/client/order/orderid | Ordered – id заказа |
| Получение списка своих заказов | get | /api/user/client/clientid | Clientid – id клиента |
| Удаление учетной записи | delete | /api/user |  | Доступен авторизованным пользователям и администраторам кафе |
| Получение информации об администраторе | get | /api /user/admin |  | Доступен авторизованным администраторам кафе |
| Создание одного кафе | post | /api/cafe |  |
| Изменение информации о кафе | put | /api/cafe/cafeid | Cafeid – id кафе |
| Удаление кафе | delete | /api/cafe/cafeid | Cafeid – id кафе |
| Обновление списка столиков кафе | post | /api/cafe/cafeid/table | Cafeid – id кафе |
| Обновление изображения плана помещения кафе | post | /api/cafe/cafeid/worktableimg | Cafeid – id кафе |
| Получение списка заказов столиков в кафе | get | /api/user/admin/cafeid | Cafeid – id кафе |
| Изменение изображения кафе | post | /api /cafe/cafeid/image | Cafeid – id кафе |

Для того, чтобы назначить каждому пути URL и типу HTTP-запроса свой метод, используется механизм роутинга Express. Например, так выглядит назначение метода создания кафе:

router.post('/cafe', auth, cafeCtrl.cafesCreate);

Название функции – это название HTTP-запроса (post), первый параметр функции – URL, второй – функция промежуточного обработчика auth, в этом приложении она отвечает за проверку права доступа, третий параметр – функция обработки запроса.

### Описание методов API

#### Получение списка кафе

Используется на главной странице web-сервиса для отображения списка кафе. Метод возвращает объект с полной информацией обо всех кафе.

#### Получение подробной информации о кофе

В качестве GET-параметра получает id кафе, по этому id ищется в базе данных одно заведение. Метод возвращает полную информацию о нём.

#### Регистрация

Получает в POST-запросе имя, email, пароль и тип пользователя. Если какого-то значения нет, возвращается ошибка. Имя, email и тип пользователя сохранятся напрямую, а пароль преобразуется в хеш.

Чтобы сделать хеш, сначала нужно сгенерировать соль. Соль – это строка с шестнадцатеричным представлением случайного 16 байтного числа. С помощью специализированной библиотеки crypto из значения соли и пароля генерируется хеш. Именно хеш и соль сохраняются в базу.

В зависимости от того, какой тип пользователя был указан при регистрации, создаётся запись в таблице Admins или Clients с id созданного пользователя.

При успешном создании записей в обеих таблицах генерируется токен из имени, email и типа пользователя сроком действия 7 дней. Он возвращается в качестве результата успешной регистрации. Таким образом, зарегистрировавшийся пользователь сразу авторизуется в системе.

#### Авторизация

Для авторизации пользователь должен указать в запросе email, паспорт и тип пользователя. Если какого-то из значений нет, возвращается ошибка. Для авторизации используются возможности библиотеки Passport, при успешной аутентификации метод возвращает токен.

#### Получение информации о схеме помещения кафе

Получает в GET-параметре id кафе. По этому id метод делает запрос к таблице Cafes, чтобы найти изображение схемы помещения кафе. Затем, в таблице Tables ищутся все столики, расположенные в этом кафе. Информация о столиках и изображение помещаются в JSON-объект и возвращаются в качестве результата.

#### Получение краткой информации о заказах в кафе

Для того чтобы сделать заказ, нужна информация о времени начала и конца каждого заказа и столик, который забронировали, не нужна информация о том, кто сделал заказ. Метод получает в GET-параметре id кафе, по нему в базе находятся все заказы в этом кафе и возвращаются в качестве результата.

#### Получение информации о пользователе

Метод доступен только авторизованным посетителям кафе. И возвращает изображение кафе. Ему не нужно передавать id пользователя, так как он берёт эту информацию из поля payload, добавляемую в функции промежуточного обработчика auth.

#### Изменение фотографии пользователя

В POST-параметре передаётся изображение, для его сохранения используется библиотека 'multiparty'. Код для использования библиотеки и сохранения файла изображения вынесен в отдельную функцию под названием saveImg(). В качестве параметров в неё передаются объект запроса и callback, вызываемый в случае успешного сохранения файла. Изображения хранятся в директории «/upload». В качестве callback`а предаётся метод сохранения изображения в базу данных, а именно строки с именем изображения.

Метод доступен только авторизованным посетителям кафе. Для поиска в таблице Clients записи посетителя используется поле «payload» из тела запроса.

#### Создание отзыва о кафе

Для сохранения отзыва нужна запись из таблицы Cafes, для ее нахождения используется id кафе, переданного в GET-параметре. Создается объект отзыва и помещается в массив reviews записи о кафе.

Значение рейтинга кафе – это среднее значение оценки всех отзывов о кафе. Поэтому, при добавлении отзыва нужно пересчитать новый рейтинг кафе. В случае успеха, метод возвращает объект отзыва.

#### Удаление отзыва

В GET-параметре передаются значения id кафе и id отзыва. При удалении отзыва также нужно пересчитать значение рейтинга кафе. В случае успеха возвращается код ответа сервера 204 и null.

#### Создание заказа

Метод доступен только авторизированным посетителям web-сервиса. Создаётся новая запись в таблице Orders и добавляется один балл в поле order соответствующей записи о кафе. Возвращает объект заказа.

#### Удаление заказа

Метод доступен только авторизированным посетителям web-сервиса. Удаляется запись в таблице Orders. В случае успеха возвращается код ответа сервера 204 и null.

#### Получение списка своих заказов

Метод доступен только авторизованным посетителям кафе. Для каждого заказа нужны следующие поля: номер столика, количество мест за столиком, начало заказа, конец заказа, название кафе. Для этого нужно сделать выборку из таблиц Tables, Orders и Cafes.

#### Удаление учетной записи

Метод доступен только авторизованным посетителям кафе. При удалении учетной записи используется поле «payload» тела запроса. По id, указанному там, находятся и удаляются соответствующие записи в таблице Users и в одной из таблиц Clients и Admins, в зависимости от типа пользователя. При успешном удалении возвращается сообщение "Пользователь удалён".

#### Получение информации об администраторе

Метод доступен только авторизованным администраторам кафе. И возвращает id кафе, созданного администратор. Ему не нужно передавать id пользователя, так как он берёт эту информацию из поля payload, добавляемую в функции промежуточного обработчика auth.

#### Создание одного кафе

Метод доступен только авторизованным администраторам кафе. Если на администратора уже зарегистрировано кафе, то метод возвращает ошибку "Этот пользователь уже создал кафе". Иначе, сохраняется новая запись в таблицу Cafes и в поле cafeid таблицы Admins сохраняется id созданного кафе. Метод возвращает объект кафе.

#### Изменение информации о кафе

Метод доступен только авторизованным администраторам кафе. Обновляет информацию о кафе по id, переданному в GET-запросе. При успешном сохранении, метод возвращает объект кафе.

#### Удаление кафе

Метод доступен только авторизованным администраторам кафе. Удаляется запись о кафе в таблице Cafes и обнуляется значение cafeid в таблице Admins. В случае успеха возвращается код ответа сервера 204 и null.

#### Обновление списка столиков кафе

Метод доступен только авторизованным администраторам кафе. Сначала удаляются все столики стоящие в кафе, которые хранятся в таблице Tables, а потом в неё сохраняются новые данные о столиках. В случае успеха возвращается сообщение об успехе.

#### Обновление изображения плана помещения кафе

Метод доступен только авторизованным администраторам кафе. Для обновления изображения используется метод saveImg(). Изображение сохраняется в поле workTableImg записи в таблице Cafes. В случае успеха возвращается строка с путём до изображения.

#### Получение списка заказов столиков в кафе

Метод доступен только авторизованным администраторам кафе. Для каждого заказа нужны следующие поля: номер столика, начало заказа, конец заказа, имя заказчика. Для этого нужно сделать выборку из таблиц Tables и Users.

#### Изменение изображения кафе

Метод доступен только авторизованным администраторам кафе. Для обновления изображения используется метод saveImg(). Изображение сохраняется в поле img записи в таблице Cafes. В случае успеха возвращается строка с путём до изображения.

# Описание разработки клиентской части

## Используемые возможности Angular

В Angular есть разные средства для структуризации приложения. Для того чтобы описать клиентскую часть приложения, нужно иметь представление о них.

Директивы — это возможность Angular, позволяющая создавать новые атрибуты, чтобы добавлять новое поведение существующим HTML элементам, или новые теги, чтобы создавать новые компоненты. Директивы можно и нужно использовать для повышения модульности приложения, выделения обособленной функциональности в компоненты, в том числе и для повторного использования.

Сервисы в AngularJS представляют собой специализированные объекты или функции, выполняющие некоторые общие для всего приложения задачи. Существуют встроенные сервисы, кроме того можно создавать собственные.

Фильтры в AngularJS позволяют провести некоторую предобработку данных перед их выводом на страницу. Это позволяет хранить данные в другом виде, чем они будут выведены в представлении и отделить бизнес-логику от логики модификации данных.

В приложении используется библиотека директив Angular UI Bootstrap и библиотека для загрузки изображений FileUploader.

## Маршрутизация

На клиенте используется собственная маршрутизация для ориентации по страницам. Этот механизм убран из ядра Angular и перенесён в модуль «ngRoute», предоставляющий сервис $routeProvider. Метод $routeProvider.when принимает два параметра: строку с названием маршрута и объект с настройками маршрута. Объект задает представление (название файла шаблона) и обрабатывающий его контроллер с помощью параметров templateUrl и controller. Метод $routeProvider.otherwise() вызывается, если не было совпадения ни с одним из маршрутов. Если не было совпадения ни с одним из маршрутов, происходит редирект на страницу 404. Листинг кода с описанием маршрутов:

$routeProvider

.when('/',{

templateUrl: '/home/home.view.html',

controller: "homeCtrl",

controllerAs: 'vm'

})

.when('/cafe/:cafeid',{

templateUrl: '/cafe/cafe.view.html',

controller: "cafeCtrl",

controllerAs: 'vm'

})

.when('/register',{

templateUrl: '/auth/register/register.view.html',

controller: "registerCtrl",

controllerAs: 'vm'

})

.when('/login',{

templateUrl: '/auth/login/login.view.html',

controller: "loginCtrl",

controllerAs: 'vm'

})

.when('/dashboard',{

templateUrl: '/dashboard/dashboard.view.html',

controller: "dashboardCtrl",

controllerAs: 'vm'

})

.when('/admin',{

templateUrl: '/admin/admin.view.html',

controller: "adminCtrl",

controllerAs: 'vm'

})

.when('/404',{

templateUrl: '/404.html'

})

.otherwise({redirectTo: '/404'});

## Сервисы

### Сервис для работы с пользователями

#### Описание

Сервис называется authentication. Это сервис предназначен для регистрации, авторизации, получения информации о пользователе и его статусе. Сервис использует встроенные в Angular сервисы $http (для запросов к API) и $window (для обращений к локальному хранилищу браузера).

#### Метод saveToken()

Сохраняет токен, возвращенный сервером после авторизации пользователя, в локальное хранилище. Это позволяет хранить состояние на клиенте.

#### Метод getToken()

Читает токен из локального хранилища и возвращает его.

#### Метод login()

Получает в качестве параметра данные о пользователе (email, пароль и тип) и делает запрос к API для авторизации пользователя в приложении. В случае успеха вызывает метод saveToken() с полученным от сервера токена.

#### Метод register()

Получает в качестве параметра данные о пользователе (имя, email, пароль и тип) и делает запрос к API для регистрации пользователя в приложении. В случае успеха вызывает метод saveToken() с полученным от сервера токена.

#### Метод logout()

Удаляет токен из локального хранилища.

#### Метод isLoggedIn()

Проверяет, авторизован ли пользователь. В качестве параметра функция получает строку с типом пользователя. Читает токен, если его нет, то функция возвращает false, иначе расшифровывает часть токена с информацией о пользователе и его «сроке годности». Если тип пользователя не совпадает с тем, что был передан в функцию в качестве параметра, возвращается false, иначе проверяется, не истёк ли срок годности токена и возвращается результат.

#### Метод currentUser()

Возвращает объект с данными об авторизованном пользователе: id, имя, тип, email. Эти данные извлекаются из токена.

#### Метод deleteUser()

С помощью токена определяется, авторизован ли пользователь, если да, то делается запрос к API. В заголовке запроса передаётся поле «Authorization» со значением токена. Этот заголовок добавляется каждый раз, когда метод API требует аутентификации пользователя.

#### Метод getUserInfo()

Получение информации о посетителе кафе, метод возвращает объект запроса, к которому можно прикреплять callback`и для обработки возвращаемых сервером данных.

#### Метод getAdminInfo()

Подобен методу getUserInfo(), но предназначен для получения данных об администраторе кафе.

### Сервис для работы с информацией о кафе

#### Описание

Сервис называется cafeData. Это сервис предназначен для CRUD-операций с кафе, удаления и добавления отзывов. Сервис использует сервисы $http (для запросов к API) и authentication (для получения информации о пользователе).

#### Метод getCafeList()

Метод возвращает объект запроса к методу API для чтения списка кафе.

#### Метод getCafeById()

Метод получает id кафе, данные о котором нужно прочитать. Возвращает объект запроса к методу API для чтения данных о кафе.

#### Метод addReviewById()

Сохраняет новый отзыв о кафе. Получает id кафе и объект отзыва о кафе. Использует заголовки для подтверждения авторизации. Возвращает объект запроса к методу API.

#### Метод deleteReviewById()

Удаляет отзыв о кафе. Получает id кафе и id отзыва. Использует заголовки для подтверждения авторизации. Возвращает объект запроса к методу API.

#### Метод updateCafe()

Обновляет данные о кафе. Получает id кафе и объект данных о кафе.

#### Метод deleteCafe()

Удаляет кафе. Получает id кафе.

#### Метод addCafe()

Создаёт кафе. Получает объект данных о кафе и возвращает объект запроса к методу API.

### Сервис для работы с заказами и столиками

#### Описание

Сервис называется workTableService. Это сервис предназначен для CRUD-операций с заказами, обновление данных о столиках. Сервис использует сервисы $http (для запросов к API) и authentication (для получения информации о пользователе).

#### Метод getWorkTableInfo()

Метод для чтения информации о столиках и схемы помещения кафе. Принимает в качестве параметра id кафе.

#### Метод getOrdersInfo()

Метод для чтения краткой информации о заказах. Принимает в качестве параметра id кафе.

#### Метод updateTablesList()

Обновляет информацию о столиках в кафе. Принимает в качестве параметра id кафе и данные о столиках. Для использования пользователь должен быть авторизован как администратор.

#### Метод createOrder()

Метод создания заказа, получает объект заказа. Для использования пользователь должен быть авторизован как посетитель.

#### Метод getOrderByClientId()

Чтение информации о заказах пользователя. Получает id посетителя. Для использования пользователь должен быть авторизован как посетитель.

#### Метод getOrderByCafeId()

Чтение информации о заказах в кафе. Получает id администратора. Для использования пользователь должен быть авторизован как администратор кафе.

#### Метод deleteOrder()

Удаление заказа. Получает id заказа. Для использования пользователь должен быть авторизован как посетитель.

## Директивы

### Директива для отображения оценок

Преобразует числовое значение оценки или рейтинга в графическое отображение в виде звезд (Рисунок ). Количество закрашенных звезд равно округлённому значению рейтинга. Для передачи значения рейтинга или оценки в директиву используется атрибут rating директивы. Листинг использования директивы:

<small rating-stars rating="cafe.rating" class="pull-right"></small>

Рисунок - Директива для отображения оценок

### Директива для управления авторизацией

В header`е страниц находится разные ссылки в зависимости от того, авторизован ли пользователь и какая роль у пользователя. Неавторизованный пользователь видит ссылки на страницы авторизации и регистрации. Администратор кафе видит своё имя с ссылкой на панель администратора и ссылку на выход из приложения – удаление данных о авторизации и редирект на главную страницу приложения. Посетитель кафе видит то же самое, но вместо ссылки на панель администратора – ссылка в личный кабинет.

В контроллере директивы используются сервисы $location (для редиректа в случае выхода из приложения) и authentication (для получения данных об авторизации и пользователях).

Рисунок - Директива для управления авторизацией, если пользователь не авторизован

Рисунок - Директива для управления авторизацией, если пользователь авторизован как посетитель

Рисунок - Директива для управления авторизацией, если пользователь авторизован как администратор кафе

### Директива конструктора схемы помещения кафе

#### Описание

Директива используется в панели администратора. В контроллере директивы используются сервисы $document (для обработки событий движения мышкой), $uibModal (для создания модальных окон), coordsService (для работы с координатами), authentication (для получения данных об авторизации и пользователях), workTableService (сервис для получения данных о заказах и столиках).

Фабрика создания объектов для работы с координатами coordsService. При работе с перемещением объектов по области нужно учитывать их положение относительно других объектов. Принимает селектор объекта, чьи координаты нужно учитывать, а также селектор объекта, относительно которого нужно высчитывать координаты. Второй параметр не обязателен.

В начале работы с директивой, её контроллер делает запрос к API для получения изображения схемы помещения и информации о столиках. Если эта информация уже есть, то изображение устанавливается в качестве фонового у рабочей области, а информация о столиках загружается в массив vm.figures, где хранятся данные обо всех столиках. Каждый столик – это объект класса Figures.

#### Панель управления

На панели находятся следующие кнопки для манипулирования схемой помещения кафе:

* Создание прямоугольного столика. Создаётся объект класса Figure, конструктору передаётся тип фигуры, чтобы задать ей нужный класс. При создании столику задаётся номер по умолчанию – место в массиве. Созданный объект помещается в массив vm.figures.
* Создание круглого столика. Аналогично созданию прямоугольного столика.
* Удаление объектов, находящихся в фокусе (одного или нескольких)
* Очистка рабочей области от столиков (в том числе фокуса)
* Обновление изображения. При нажатии открывается модельное окно, куда передается id кафе в качестве поля контроллера. После закрытия окна возвращается строка с адресом изображения. С помощью js объекта Image, узнаются высота и ширина изображения и в объект vm.workTableImg сохраняются все css-стили для отображения на рабочей области.
* Сохранение столиков. Прежде чем сохранять информацию, данные проверяются на пустоту и уникальность номеров столиков в пределах кафе. Если найдена ошибка, выводится сообщение об этом. При удачном сохранении тоже. Для проверки уникальности номера столика есть функция сравнивающая номер каждого столика со всеми другими и в случае совпадения возвращающая false.

Кроме того, справа на панели находятся формы для просмотра и изменения информации об одном выделенном столике (находящемся в фокусе): его номер и количества мест за ним.

#### Рабочая область и объекты

На рабочей области отображается изображение схемы помещения, все столики и действия над ними. Ширина и высота рабочей области равны ширине и высоте изображения схемы помещения.

В области отображаются два списка столиков: список всех столиков и список столиков, выделенных по области, внутри тега для выбора по области. С этим тегом связан специальный объект sFrame. SFrame и Figures имеют несколько методов с одинаковым название и назначением, что помогает оба этих объекта сохранять в объект для хранения выделенного столика (одного или нескольких). Внутри sFrame хранится список выделенных столиков, при добавлении туда они временно удаляются из vm.figures, после разгруппировки обратно туда возвращаются. Выделение по области происходит в следующем порядке:

1. Обработчик опускания клавиши мыши на рабочую область (не на фигуру или тег, связанный с sFrame) перемещает фокус на объект sFrame и задаёт ему нулевые значения ширины и высоты. Кроме того, создаётся объект coordsService для сохранения координат угла выделенной области, с которого началось выделение.
2. Вешаются обработчики события движения мышкой с нажатой клавишей по телу документа с помощью сервиса $document и события отпускания клавиши мыши.
3. При каждом срабатывании события движения мышкой с нажатой клавишей вызывается обработчик этого события, вычисляет координаты противоположного угла области выделения и обновляет поле объекта sFrame c css-стилями sFrameStyle. Изменения тут же отображаются в представлении.
4. При событии отпускания клавиши мыши в последний раз пересчитаваются размеры области выделения и вызывается метод для проверки попадания столиков в область. Метод проходит по массиву со столиками и, если столик полностью попадает в область, перемещает его из списка vm.figures в список sFame.children. В конце удаляются обработчики событий движения мышкой и отпускания клавиши.

Если объект sFrame теряет фокус (даже если для того, чтобы сразу снова получить его), то все столики из sFame.children возвращаются в vm.figures.

Для перемещения столиков и выделенной области используется один и тот же механизм:

1. Обработчик опускания клавиши мыши на элемент перемещает фокус на этот элемент, создаётся объект coordsService для работы с координатами элемента.
2. Вешаются обработчики события движения мышкой с нажатой клавишей по телу документа с помощью сервиса $document и события отпускания клавиши мыши.
3. При каждом срабатывании события движения мышкой с нажатой клавишей вызывается обработчик этого события, вычисляет координаты элемента и обновляет поле объекта c css-стилями. Изменения тут же отображаются в представлении.
4. При событии отпускания клавиши мыши в последний раз пересчитаваются координаты элемента. Удаляются обработчики событий движения мышкой и отпускания клавиши.

Если столик находится в фокусе, но у него можно изменить размер с помощью специальных управляющих элементов. Они работают похожим образом, что и перемещение элементов и выделение области, но вместо координат меняются значения ширины и высоты столика.

Для переноса фокуса вызывается метод changeFocus(), который в случае непустоты объекта vm.focusObj для фокуса убирает стиль фокусировки у текущего фокуса и вызывает его метод delete(). Затем присваивает vm.focusObj новое значение и обновляет его стили. Метод delete() обычного столика – это просто заглушка, а в случае sFrame, он переносит все столики из области обратно в массив vm.figures.

Чтобы каждый раз не приписывать значение «px» при обновлении координат и размеров фигур, был создан фильтр formatStyle, который просто добавляет это в конец всех полей объекта.

Рисунок - Конструктор схемы помещения кафе

## Страницы и их контроллеры

### Главная страница

Как уж было сказано, на главной странице выводится список кафе с краткой информацией о них. Логика контроллера этой страницы очень проста: обратиться к методу getCafeList() сервиса cafeData и поместить полученный с сервера список кафе в массив.

Массив кафе выводится в цикле на главной странице. Для каждого кафе создаётся блок с фотографией кафе, под фотографией указано название и рейтинг в виде звёзд. При наведении на блок поверх фотографии появляется другая информация о кафе: кухня, средний чек, адрес и так далее.

Слева от списка расположены инструменты для фильтрации и сортировки списка кафе. При изменении параметров фильтрации и сортировки список моментально меняется. Все 4 поля фильтров действуют на список одновременно: например, можно сортировать только заведения с оплатой через интернет, при этом кафе, где нет такого способа оплаты, не будут участвовать и их не будет видно на экране.

Эти фильтры и сортировки сделаны на основе встроенных возможностей Angular, а не логики контроллера. В цикле вывода кафе указывается, какие поля модели влияют на фильтрацию и сортировку. Эти поля зависят от формы в инструментах фильтрации и мгновенно изменяются вместе с ними. Листинг цикла вывода списка кафе:

<div class="col-md-4 cafe-preview" ng-repeat="cafe in vm.data.cafes | filter:vm.paymentFilter | filter:vm.textFilter | filter:vm.cuisineFilter | orderBy: vm.selectFilter :true">…</div>

«Filter» - для фильтрации, «orderBy» - для сортировки. Значение vm.paymentFilter отвечает за фильтрацию по способу оплаты, vm.textFilter – по значению текстового поля, vm.cuisineFilter - по кухне кафе, а в поле vm.selectFilter указан критерий по которому нужно сортировать список.

### Страница кафе

На странице отображаются все данные о кафе, кроме служебных. Фотография кафе выводится на странице в виде фонового изображения страницы. Контроллер этой страницы использует следующие сервисы: $routeParams (для получения id кафе), $location (для возвращения на страницу после авторизации), $uibModal (для открытия модального окна для добавления отзыва о кафе), cafeData (для получения информации о кафе), authentication (для получения информации о пользователях).

На странице кафе, в зависимости от авторизованности и типа пользователя, отображается немного разная информация. Посетителю кафе доступны кнопки добавления отзыва и создания заказа. Остальные пользователи видят кнопки с предложением авторизоваться в приложении. Хотя проверка права доступа к методу API происходит на сервере, это сделано для улучшения внешнего вида сайта и понятности интерфейса.

Если посетитель кафе оставил отзыв о кафе, то он может его удалить, используя кнопку в заголовке отзыва. Чужие отзывы удалять нельзя.

Рисунок – Страница кафе

### Страница регистрации

На этой странице находится форма регистрации с полями имя, email, пароль и тип пользователя. Все поля должны быть заполнены, иначе при попутке отправить форму выводится сообщение об ошибке.

Если пользователь уже авторизован, вместе формы ему видна только надпись «Вы уже вошли в приложение».

Рисунок Форма регистрации

### Страница авторизации

На этой странице находится форма авторизации с полями email, пароль и тип пользователя. Все поля должны быть заполнены, иначе при попутке отправить форму выводится сообщение об ошибке.

Если пользователь уже авторизован, вместе формы ему видна только надпись «Вы уже вошли в приложение».

Рисунок Форма авторизации

### Личный кабинет пользователя

На странице личного кабинета выводится информация о пользователе (имя, email, фото), список заказов, принадлежащих пользователю и возможность изменения и удаления пользователя.

На странице открываются два модальных окна: для подтверждения удаления профиля и для обновления фотографии пользователя. В случае удачного удаления, пользователь выходит из приложения и попадает на главную страницу. При удачном обновлении фото, возвращается его адрес и обновляется фото в представлении.

В списке заказов отображается название кафе, номер столика и количество мест за ним и время начала и конца заказа. Кроме того, есть возможность удалять заказы.

Рисунок – Личный кабинет пользователя

### Панель администратора кафе

Панель администратора структурирована с помощью вкладок. В каждой вкладке находится одна опция управления кафе:

* Заказы
* План помещения
* Информация о кафе
* Изображение

Если администратор ещё не создал кафе, под заголовком страницы находится сообщение о необходимости его создать. Кроме того, во всех вкладках, кроме «Информация о кафе», недоступен функционал.

Контроллер использует сервисы $uibModal (модальное окно для изменения изображения кафе), cafeData (для чтения данных о кафе), authentication (Для чтения данных об администраторе и проверки типа пользователя), workTableService (для чтения данных о заказах). При загрузке страницы делаются запросы к API для загрузки данных о кафе и данных о заказах.

Если кафе ещё не создано, то во вкладке «Информация о кафе» отобржается пустая форма, а поля в объекте vm.cafeForm с информацией о кафе заполняются начальными значениями. Поле vm.cafeForm.contacts должно быть пустым массивом, а vm.cafeForm.timetable – массивом с 7 объектами, для создания расписания кафе. Виды кухни и способы оплаты задаются массивом предопределенных значений.

Для добавления контактов используется группа полей: тип контакта, название и значение. Существуют контакты в двух типов: в виде текста (для номера телефона, адреса офиса и т.п.) и в виде ссылки (для ссылок на соц. сети или сайты). Есть возможность добавлять произвольное количество контактов. После добавления контакта он отображается в списке чуть ниже группы полей. От туда его можно удалить с помощью кнопки с изображением крестика с левой стороны элемента списка.

Похожий механизм используется и для добавления расписания работы кафе. Доступные поля: день недели (мультиселект со значениями дней недели из vm.cafeForm.timetable), время открытия и закрытия (тип «time» HTML-тега input). Для каждого дня недели (для одного или сразу для нескольких) можно задать время открытия и закрытия кафе. Пропущенные дни недели считаются нерабочими. Как и на элементе добавления контактов, после добавления расписания на день, оно отображается в списке в элементе формы и его можно удалить. При добавлении сразу нескольких дней недели каждый день добавляется в список отдельно. При добавлении времени открытия и закрытия дню недели в объекте в массиве vm.cafeForm.timetable появляются соответствующие значения.

Рисунок Вкладка «Информация о кафе»

Во вкладке «План помещения» выводится директива конструктора схемы помещения кафе.

Во вкладке «Заказы» выводится 3 списка с заказами: будущие заказы, заказы на сегодня и архив заказов. Для разделения заказов по этим трём группам создан фильтр «dateFilter». Он принимает объект (массив заказов) и флаг, обозначающий, какое подмножество заказов нужно получить. При отсутствующем флаге фильтр возвращает только сегодняшние заказы. При флаге false – архивные заказы, при true – будущие заказы. В этом фильтре используется библиотека moment.js для сравнения дат.

Рисунок Вкладка «Заказы»

Во вкладке «Изображение» отображается текущее изображение кафе и кнопка для вызова модального окна обновления изображения. При удачном изменении, после закрытия окна возвращается адрес нового изображения, и обновляется изображение на странице.

Рисунок Вкладка «Изображение»

### 404

Эта страница не содержит никакой логики, только уведомление о том, что запрошенная страница не существует, и ссылку на главную страницу. Носит исключительно справочный характер.

Рисунок Страница 404

## Модальные окна и их контроллеры

### Окно заказа столика

Это модальное окно предназначено для совершения заказа посетителями кафе. Оно открывает на странице кафе по нажатию на соответствующую кнопку. В этом окне отображается схема помещения кафе, элемент для выбора даты и слайдер с двумя бегунками для выбора времени начала и конца бронирования столика. Для совершения заказа нужно выбрать значения всех трёх элементов управления.

Элемент для выбора даты представляет собой текстовое поле с кнопкой справа, предназначенной для открытия календаря.

Рисунок Окно заказа столика

### Окно обновления изображения

Модальные окна для обновления изображений в разных частях приложения отличаются только методом API. В модальном окне находится форма с полем выбора изображения. Для отправки на сервер изображений используется модуль Angular FileUploader. В случае успешной отправки, окно возвращает данные ответа, а именно строку с названием изображения.

### Окно для написания отзыва о кафе

Форма добавления отзыва (Рисунок 2.12) находится в модальном окне на странице кафе и появляется после нажатия кнопки «Добавить отзыв». Форма содержит поля формы оценка и комментарий, оба поля обязательны для заполнения. После отправки отзыва, окно закрывается, а в списке отзывов появляется новый отзыв.

Рисунок 2.12 – Модальное окно с формой добавления отзыва

### Окно подтверждения удаления пользователя

Это простое модальное окно для подтверждения действия с текстом «Вы уверены, что хотите удалить ваш аккаунт?». Если пользователь подтверждает действие, то контроллер окна обращается к функции deleteUser() сервиса authentication. В случае успеха пользователь выходит из приложения и перенаправляется на главную страницу.

# Заключение

# Список литературы