# **Спецификация требований**

*[Документ в целом состоит из двух основных разделов: введения и самих требований. Но структуру разделов и подразделов можно менять, в зависимости от того как удобнее и принято в конкретном проекте. Например, можно выделить отдельно функциональные требования в один раздел, требования к интерфейсам — в другой, а остальные нефункциональные требования — в третий, как сделано в этом шаблоне. Аналогично, можно раздел раздел введения на несколько.]*

## **Введение**

*[Здесь кратко описывается назначение документа (не системы), описывается соглашения, принятые в документе, используемые нотации и так далее. Можно также описать для какой аудитории предназначен документ и каким типам читателей какие разделы могут быть особенно полезны].*

Этот документ создан с целья ознакомления с системой веб-сайта pizzeria.com. Используются нотации BPMN, описывающая бизнес процессы, и UML, описывающая дизайн системы. Для каждой графической нотации есть ссылка на источник. Документ предназначен для потребителей, поставщиков и специалистов программного обеспечения. Потребителям и поставщикам будет полезен раздел описания системы, специалистам раздел требований.

BPMN диаграмму можно посмотреть по ссылке:

<https://drive.google.com/file/d/1vVqn4flfZJ3XoCBIYy1UcioocCNWZcct/view?usp=sharing>

Описание BPMN:

https://docs.google.com/document/d/1naDU\_MkLAVSIz6TArJFvHeft8I15lh947RXeXvqS1EM/edit

## **Описание системы**

### **Назначение**

*[Назначение системы, для бизнес-задачи она решает]*

Система веб-сайта пиццерии pizzeria.com предназначена для построения процесса оформления, оплаты и доставки заказа пользователю. Система является полностью самодостаточной и независимой.

### **Границы системы и бизнес-требования**

*[Краткое описывает, что система будет делать — что будет в границе её зоны ответственности — определяет, какие бизнес-цели система предназначена реализовывать. При необходимости — можно отдельно указать, что находится вне границ системы.]*

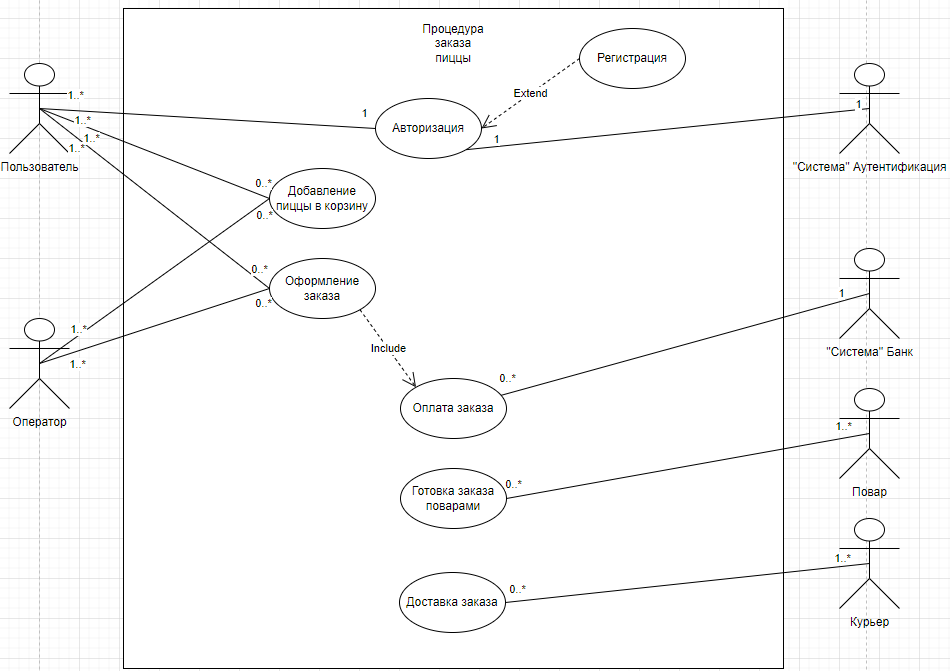
Система должна обеспечить в границах:

* Увеличение конверсию трафика до 25% за 6 месяцев.
* Уменьшение время отклика сайта до 1 секунды.
* Уменьшение время обработки оплаты с карт пользователей до 3 секунд.
* Улучшить опыт пользователей для увеличения лидов на 30%.
* Увеличить автоматизацию оформления и оплаты заказов на сайте на 50% за 3 месяца.
* Увеличить звонки операторам на 10%.
* Увеличить кол-во посетителей сайта на 60% за 6 месяцев.
* Оптимизировать время готовки заказа поварами до ~ 10 минут.
* Увеличение кол-ва добавления заказа в корзину на 50% за полгода.
* Уменьшить кол-во брошенных корзин на 80% в течении года.
* Уменьшить среднее время доставки заказа пользователю до 10 минут в черте 5 километров от заведения, до 20 минут в черте 10 километров и 30 минут в черте 15 километров.
* Увеличить кол-во зарегистрированных пользователей на 50% за 3 месяца.
* Уменьшить среднее время аутентификации до 2 секунд.

Ответственность вне границ системы:

* Улучшение скорости интернета до 300 mbps.
* Улучшение синхронизации с внешними системами банков пользователей.

[Диаграмма сценариев]



https://app.diagrams.net/#G1mQIlx-\_8R7i9CHn3CW75rVYEg8IIQqgP

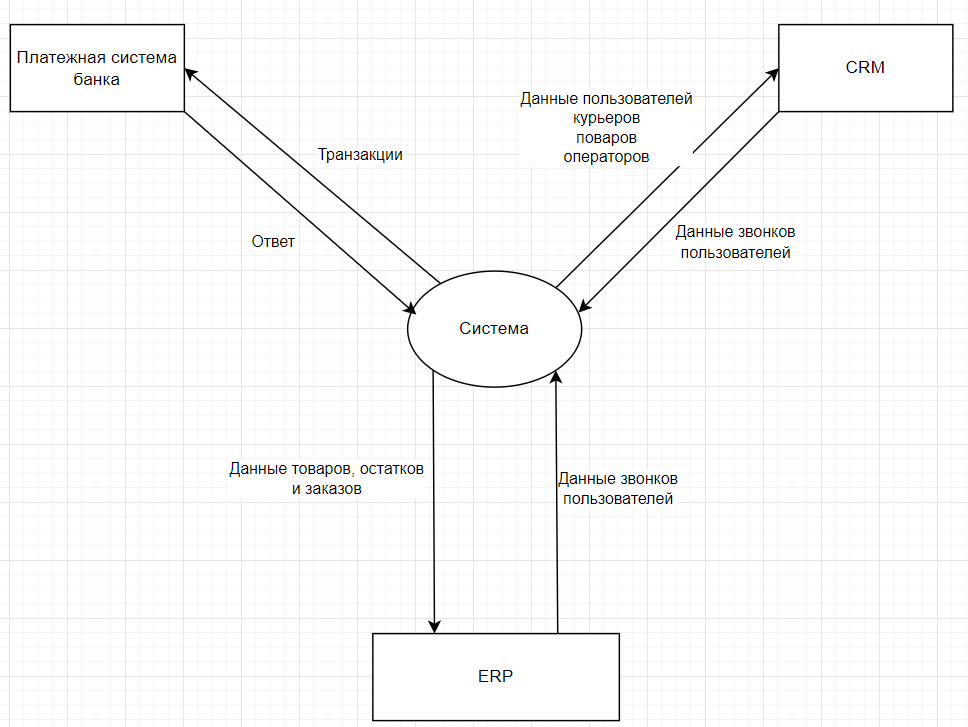
### **Краткое описание контекста использования**

*[Кратко перечисляет какие API и пользовательские интерфейсы система предоставляет и использует, какое аппаратное обеспечение нужно для её работы. Детальное описание требований к программным и пользовательским интерфейсам будет дальше.]*

Система взаимодействует с внешними системами:

* Предоставляет данные пользователей, курьеров, поваров и операторов CRM-системе посредством её REST API.
* Предоставляет данные заказов, продуктов и остатков ERP-системе посредством REST API.
* Принимает информацию о звонках пользователя CRM- и ERP-системами посредством REST API.
* Предоставляет пользовательские банковские данные платежной системе банка пользователя и получает ответ через SOAP API банка.

[Контекстная диаграмма]



https://app.diagrams.net/#G1mQIlx-\_8R7i9CHn3CW75rVYEg8IIQqgP

### **Заинтересованные лица**

*[Описание заинтересованных лиц и их роль в проекте.]*

**Внутреннее**:

**Сторонники**:

Заказчик/Спонсор/Владелец

Проект менеджер

Дизайнер

Тестировщик

Фронтенд

Бэкэнд

Системный аналитик

Копирайтер

Инженер QA

**Внешние**:

**Сторонники**:

Роскомнадзор

Платформа для онлайн коммерции

Платежный шлюз

**Противники**:

Конкуренты

Ссылка на роли ЗЛ:

<https://docs.google.com/spreadsheets/d/1VMam_SLy7_QiFzFMtFYwDpM7D00ddgQSsAwqhnrgjGg/edit#gid=980223765>

### **Типы пользователей**

*[Кратко перечисляет типы пользователей, требования которых система будет реализовывать, с их характеристиками.]*

1. Пользователь
   * Выбирает и оплачивает продукт
   * Опытный пользователь приложений и веб-сайтов
   * От среднего до высшего образования
   * Низкая техническая грамотность
2. Курьер
   * Принимает заказ у кухни и доставляет до пользователя
   * Опытный пользователь приложений и веб-сайтов
   * От среднего до высшего образования
   * Низкая техническая грамотность
3. Повар
   * Принимает и готовит заказ
   * Опытный пользователь приложений и веб-сайтов
   * От среднего до высшего образования
   * Низкая техническая грамотность
4. Оператор
   * Принимает звонки и оформляет заказы
   * Опытный пользователь приложений и веб-сайтов
   * От среднего до высшего образования
   * Низкая техническая грамотность
5. Админ
   * Следит и помогает с проблемами системы
   * Опытный пользователь приложений и веб-сайтов
   * От среднего до высшего образования
   * Высокая техническая грамотность

### **Основные функции**

*[Кратко перечисляет основные функции системы с точки зрения пользователей — фактически, пользовательские требования. Здесь можно использовать информацию из пользовательских историй или перечислить пользовательские сценарии использования]*

1. Я как новый пользователь хочу создать реквизиты для входа, чтобы иметь доступ к услугам сайта
2. Я как новый пользователь хочу активировать учетную запись, чтобы пользоваться услугами сайта
3. Я как пользователь хочу видеть все доступные для заказа пиццы, чтобы выбрать понравившуюся
4. Я как пользователь хочу выбирать топпинг на мою пиццу, чтобы собрать пиццу на свой вкус
5. Я как пользователь хочу проверить свой заказ в корзине, чтобы убедиться, что заказ правильный
6. Я как пользователь хочу вести свой адрес, чтобы заказ был доставлен точно в мой дом
7. Я как пользователь хочу выбрать способ оплаты, чтобы было удобно расплачиваться
8. Я как пользователь хочу просматривать мой статус заказа, чтобы знать когда ожидать пиццу
9. Я как оператор хочу помочь с выбором пиццы, чтобы пользователь мог определиться с пиццей
10. Я как оператор хочу помочь с выбором топпинга, чтобы собрать пиццу под вкус пользователя
11. Я как оператор хочу собрать адрес и контактную информацию пользователя, чтобы было известно куда доставить заказ
12. Я как оператор хочу проверить правильность состава заказа, чтобы подтвердить состав заказа
13. Я как оператор хочу зафиксировать способ оплаты от пользователя в системе, чтобы курьер знал как будет проходить способ оплаты
14. Я как оператор хочу отправить статус заказа, чтобы пользователь знал когда ожидать пиццу
15. Я как повар хочу видеть все имеющиеся заказы на данный момент, чтобы знать какую пиццу готовить следующим
16. Я как повар хочу принять заказ на пиццу, чтобы начать готовить заказ
17. Я как повар хочу собрать заказ, чтобы отдать курьеру на доставку
18. Я как курьер хочу просматривать статусы заказов, чтобы знать когда заказ готов к доставке и как скоро его надо доставить
19. Я как курьер хочу взять заказ на доставку, чтобы отдать пиццу заказчику
20. Я как курьер хочу подтверждать доставку заказа, чтобы закрыть заказ
21. Я как курьер хочу принять оплату, чтобы закрыть заказ
22. Я как пользователь хочу авторизоваться, чтобы просматривать свою историю заказов и трат на них
23. Я как пользователь хочу просматривать свою история заказов, чтобы иметь представление о своих заказах и их стоимости
24. Я как пользователь хочу сохранять пиццы в избранное, чтобы быстро и просто их заказывать

Ссылка на пользовательские требования:

https://skillfactroy.atlassian.net/jira/software/projects/TPS/boards/1

Ссылка на пользовательские сценарии:

https://skillfactroy.atlassian.net/wiki/spaces/TPS/pages

### **Ограничения**

[Основные ограничения, накладываемые на систему или её разработку внешней средой: технические ограничения, ограничения регуляторов, лицензионные ограничения и т. д.]

* Пользователи будут использовать систему, используя канал связи низкой пропускной способности.
* При разработке системы запрещается использовать open-source-компоненты.
* Система должна хранить данные пользовательских действий в течение 30 дней по требованиям федерального закона.
* Система должна работать на локальных хостах и серверах.
* Домейн должен быть локальным.
* Лицензионный софт CRM & ERP отечественного производства.
* Совместимость с браузерами

### **Определения**

*[Глоссарий терминов и расшифровка аббревиатур. Эти термины должны использоваться дальше по всему документу.]*

| Авторизация | Предоставление определённому лицу или группе лиц прав на выполнение определенных действий |
| --- | --- |
| Админ | Человек и администратор веб-сайта, в ведении которого находится обеспечение нормальной работы серверов и рабочей сети компьютеров организации путем поиска возникающих проблем, настройки, отладки, переустановки и пр. |
| База данных | Совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования, независимая от прикладных программ. Обращение к базам данных осуществляется с помощью системы управления базами данных (СУБД) |
| Балансировщик | Сервис, который перенаправляет запрос от клиента в наименее загруженный экземпляр приложения |
| Брокер сообщений | Компонент, обеспечивающий асинхронное взаимодействие между сервисами |
| Веб-сайт | Совокупность веб-страниц, объединенных по смыслу, имеющих общую структуру и навигацию |
| Веб-сайт пиццерии | Веб-сайт сети быстрого питания ‘пиццерия’ в сети Интернет |
| Веб-сервер | Компьютерное программное обеспечение и базовое оборудование, которое принимает запросы через HTTP или его безопасный вариант HTTPS. Также выступает в роли балансировщика |
| Заказ | Запрос пользователя на веб-сайте на приготовление и доставку пиццы пользователю |
| Интерфейс | Взаимодействие между отдельными системами, обеспечивающими передачу информации между пользователем-человеком и программно-аппаратными компонентами компьютерной системы |
| Клиент | Приложение или браузер, отправляющее запросы серверу с целью получения той или иной информации |
| Курьер | Человек, выполняющий доставку заказа пользователю |
| Кухня | Отдельное помещение с печью, плитой для приготовления пищи |
| Кэш | Компонент, который хранит данные, чтобы будущие запросы на эти данные могли обслуживаться быстрее |
| Логин | Персональное имя для авторизации на веб-сайте |
| Метод | CRUD методы для работы с базами данных и REST API |
| Оператор | Человек, помогающий оформить заказ пользователю |
| Оформление заказа | Процесс создания запроса на веб-сайте на приготовление и доставку пиццы пользователю |
| Пароль | Секретный набор символов, который защищает учетную запись пользователя |
| Первичная регистрация | Регистрация пользователя в системе в первый раз с вводом пользовательских данных |
| Пицца | Основной вид продукта доступный на веб-сайте для заказа и потребления |
| Пиццерия | То же, что и веб-сайт пиццерии |
| Повар | Человек, готовящий заказ пользователя |
| Пользователь | Человек, использующий систему для решения стоящих перед ним задач. Также пользователями могут выступать повара, курьеры и операторы при входе в систему. |
| Почта | Электронный адрес почты пользователя, курьера, повара и оператора |
| Продукт | Вид товара предложенный на сайте для удовлетворения потребности пользователя |
| Ресурс | Различные категории информации возвращаемые API |
| Сервер | Часть компьютерного оборудования или программного обеспечения, которое обеспечивает функциональность для других программ или устройств, именуемые клиентами |
| Сервис | Процессы обслуживания объектов Интернет. Сервисы предоставляются пользователям, программам, системам, уровням, функциональным блокам |
| Сессия в системе | Время от присоединения пользователя к серверу до обрыва связи или отсоединения. |
| Ссылка | Это прямой путь перехода от одной страницы к другой в пределах одного веб- сайта |
| Статус заказа | Состояние заказа на текущий момент |
| Страница | Документ, снабженный уникальным адресом, который можно открыть и посмотреть с помощью браузера |
| Транзакция | Логически завершенная банковская операция |
| Топпинг | Ингредиент пиццы, придающий ему вкусовую и эстетическую завершенность |
| Учетная запись | Хранимая в системе совокупность данных о пользователе, необходимая для его аутентификации и авторизации |
| Экран | То же, что и интерфейс |
| Cart | Страница, на которую пользователь отправляет продукты, чтобы затем оформить заказ на них |
| Chef | То же, что и повар |
| CSS | Язык таблицы стилей, используемый для описания представления документа, написанного на языке разметки, таком как HTML |
| Deliveryman | То же, что и курьер |
| DOM | Кросс-платформенный и независимый от языка интерфейс, который обрабатывает документ XML или HTML как древовидную структуру |
| Enumiration | Тип данных, состоящий из набора именованных значений, называемых элементами, членами, перечислением или перечислительными |
| ID | Имя, которое идентифицирует либо уникальный объект, либо уникальный класс объектов в системе |
| JavaScript Web API | Конструкции, встроенные в браузер, построенные на основе языка JavaScript |
| Operator | То же, что и оператор |
| Order | То же, что и заказ |
| Payment | Оплата за продукт через банковские транзакции |
| Product | То же, что и продукт |
| Stock | Запасы продуктов, еще не распроданных |
| Topping | То же, что и топпинг |
| User | То же, что и пользователь |
| Username | То же, что и логин |
| XBL | Язык разметки на основе XML, используемый для объявления поведения и вида XUL-виджетов и элементов XML |
| XML | Язык разметки и формат файла для хранения, передачи и восстановления произвольных данных |
| WCAG 2.0 | Руководство по обеспечению доступности веб-контента |

## **Требования**

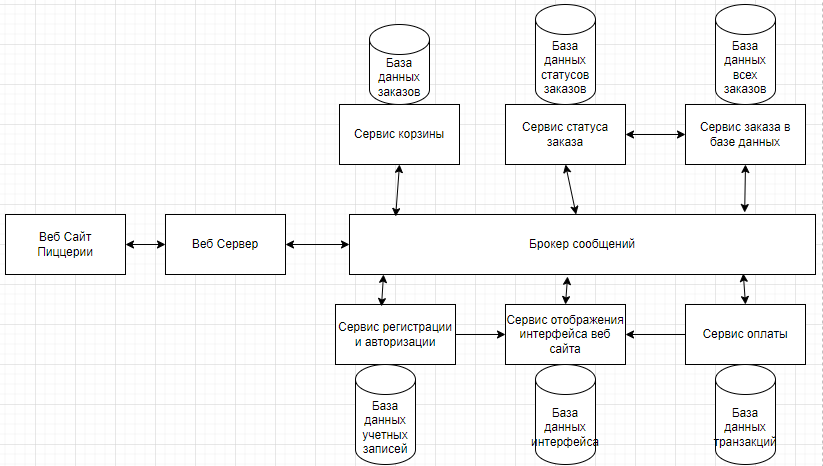
*[Основной раздел документ, содержащий системные требования. Требования (кроме сценариев) стоит формулировать в стандартной форме: [Условие] [субъект] должен [действие] [объект] [ограничение] [значение ограничения]]*

| Степень необходимости | Требования |
| --- | --- |
| **Обязательные** | 1. Когда пользователь вводит свои данные, система должна обрабатывать данные пользователя 2. Когда пользователь ввел свои данные, система должна регистрировать пользователя 3. Когда открывается главная страница пиццерии, экран должен отображать все пиццы на главной странице 4. Когда пицца выбрана, система должна предоставить возможность выбрать топпинг на пиццу для пользователя 5. В окне выбора топингов, экран должен отображать все доступные топпинги для пиццы и параметры выбранной пиццы 6. Когда продукт со всеми параметрами выбирается, система должна сохранять заказ в корзине для пользователя 7. Когда продукт со всеми параметрами удаляется, система должна удалить заказ из корзины 8. Во время оплаты заказа, интерфейс должен отображать экраны с составом заказа, способом оплаты и данными пользователя 9. Когда происходит платеж картой онлайн, система должна обрабатывать платеж пользователя через банк 10. Когда происходит платеж картой по терминалу, система должна обрабатывать платеж пользователя через банк 11. Когда происходит платеж наличными, система должна создать чек на пиццу 12. Когда происходит платеж переводом, система должна обрабатывать платеж пользователя через банк 13. Когда пользователь оплачивает заказ, система должна открыть страницу проверки одноразового пароля от банка пользователя 14. Когда заказ оформлен, система должна создать заказ в базе данных заказов 15. Когда проходит успешный платеж, система должна оформить заказ 16. Когда происходит выбор пользователем платежа наличными или по терминалу, система должна оформить заказ 17. Когда пользователь выбирает другой адрес, система должна сменить текущий адрес на выбранный 18. Когда пользователь редактирует свой номер телефона, система должна сменить телефон на выбранный 19. Когда открывается страница корзины, экран должен отображать стоимость и кол-во продуктов в корзине 20. Когда открывается страница статуса заказа, интерфейс должен отображать экраны с составом заказа, адресом, временем доставки 21. Когда заказ оформлен, система должна передать заказ на кухню для готовки 22. Когда повар принимает заказ, система должна предоставить возможность сортировки заказов по номерам, времени принятия заказа в работу, сложности, размеру 23. Когда заказ собран, система должна передать заказ на доставку курьеру 24. Когда статус заказа меняется на доставлен, система должна закрыть заказ в базе данных заказов 25. Когда пользователь вводит свои логин и пароль, система должна подтвердить логин пользователя 26. Когда пользователь выбирает другой метод авторизации, система должна изменить метод авторизации на выбранный 27. Когда открывается страница пиццерии с авторизованным пользователем, экран должен отображать возможность добавления в избранное пицц 28. Система должна получать статус заказ с пиццерии, когда у заказа меняется статус 29. Система должна отправлять статус заказа в пиццерию, когда у заказа меняется статус 30. Система должна отправлять СМС с информацией о статусе заказе пользователю, когда у заказа меняется статус |
| **Желательные** | 1. Когда пользователь завершает ввод своих данных, экран должен отображать сообщение с благодарностью за регистрацию и об отправке письма на почту с ссылкой 2. Когда открывается страница пиццерии, экран должен отображать ссылку на регистрацию 3. Когда открывается страница пиццерии, экран должен отображать ссылку на личный кабинет 4. Когда открывается страница пиццерии, экран должен отображать ссылку на авторизацию 5. Когда открывается страница пиццерии, экран должен отображать ссылку на избранное 6. Когда открывается страница пиццерии, экран должен отображать ссылку на корзину 7. Во время первичной регистрации, экран должен содержать поля для вводимых данных пользователем по адресу почты, паролю, имени, дате рождения и телефону 8. Когда регистрация окончена, система должна отправить сообщение со ссылкой для активации аккаунта на почту пользователя 9. Когда пользователь переходит по ссылке на почте, система должна перенести пользователя на главную страницу пиццерии авторизированным 10. Когда способ оплаты выбран, экран должен отображать способ оплаты в статусе заказа 11. Система должна отображать статус заказа на экране, когда у заказа меняется статус 12. Когда пользователь переходит по ссылке в личный кабинет, интерфейс должен отображать экраны с данными пользователя, историей заказов и статусом заказа 13. Во время ввода пользовательского адреса, экран должен содержать поле для вводимых данных пользователем по адресу 14. Когда пользователь добавляет пиццу в избранное, система должна сохранить данные о пицце для последующих заказов 15. Когда пользователь переходит по ссылке избранное, экран должен отображать все избранные пиццы пользователем 16. Когда пользователь проходит авторизацию, экран должна отображать поля для ввода данных 17. Когда пользователь переходит по ссылке своих историй заказов, экран должен отображать всю историю заказанных пицц с итоговой стоимостью каждой 18. Когда пользователь вводит свой номер телефона, система должна отправить одноразовый СМС код на телефон пользователя для авторизации |
| **Необязательные** | 1. Когда открывается страница пустой корзины, экран должен отображать сообщение с просьбой о добавлении пиццы в заказ |

*[Также в раздел можно включать подразделы, описывающие архитектурные решения, если проект находится уже на стадии проектирования:*\_

* *архитектурные принципы*
* *основные компоненты*
* *потоки данных между компонентами*
* *распределение функций между компонентами]*

Для легкой модификации и развития системы была выбрана микросервисная архитектура.



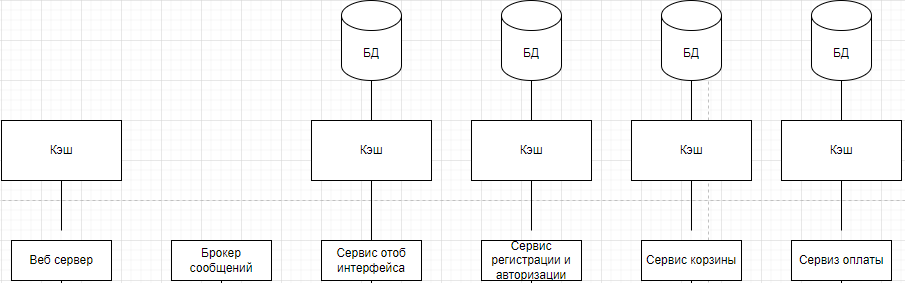
Основные компоненты микро сервисной архитектуры:

* Сервис корзины
* Сервис статуса заказа
* Сервис заказа в БД
* Сервис регистрации и авторизации
* Сервис отображения интерфейса веб-сайта
* Сервис оплаты

1. Сервис корзины реализует требования к отображению продуктов в корзине.
2. Сервис статуса заказа реализует требование отображения статуса заказа
3. Сервис регистрации и авторизации реализует требование к отображению экрана регистрации и авторизации
4. Сервис отображение интерфейса реализует требование к отображению экранов с составом заказа, способом оплаты, данными пользователя, адресом, временем доставки и историей заказов
5. Сервис оплаты реализует требование к отображению экрана оплаты заказа
6. Веб-сервер реализует требование к производительности системы
7. Брокер сообщений реализует требование к надежности системы
8. Базы данных реализуют требование к производительности и доступности системы
9. Кэш сервис реализует требование к надежности и производительности системы

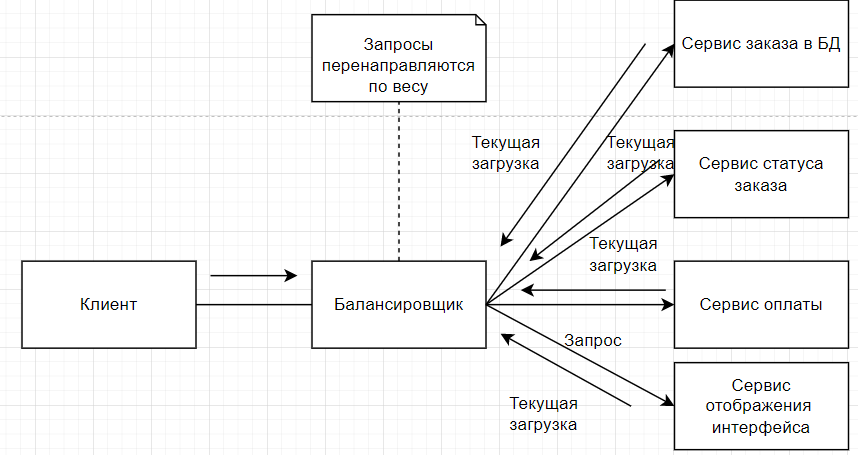
Потоки данных происходят преимущественно через брокер сообщений для асинхронного сообщения между компонентами. Однако, некоторые компоненты взаимодействуют непосредственно друг с другом.

Каждый сервис со своими основными функциями.

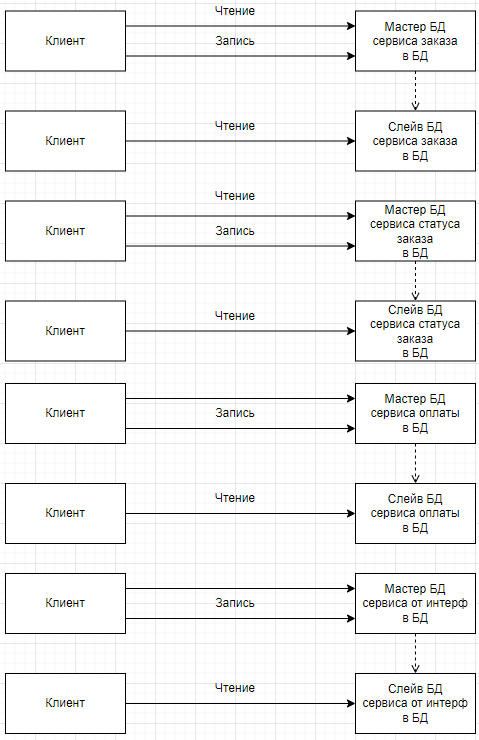


Используется многоуровневое кеширование. Также у всех сервисов, за исключением сервисов статуса заказа и заказа в БД, есть свои кэш сервисы.

Есть несколько методов работы кэша. Ссылка на описание:  
  
<https://docs.google.com/document/d/105mflhhdMPK_CCsoSS1sL5PwT5P5vQN2qFf9SKpCcos/edit>



Вертикальное масштабирование. После достижения точки возрастания затрат на поддержание и увеличение масштабирования нужен переход на горизонтальное масштабирование. Балансировка по весу подходит лучше всего.



Репликация БД для неравномерно загруженных сервисов с БД для увеличения производительности системы в целом.

Ссылка на диаграммы:

<https://app.diagrams.net/#G1mQIlx-_8R7i9CHn3CW75rVYEg8IIQqgP>

### **Сценарии использования + Функциональные требования**

*[Подразделы со сценариями использования. Их можно группировать, например по основным актором. Раздел можно объединить с функциональными требованиями, где функциональные требования будет располагаться в разделах соответствующих сценариев.]*

Новый пользователь:

1. Регистрация учетной записи
   1. Когда пользователь вводит свои данные, система должна обрабатывать данные пользователя
   2. Когда пользователь ввел свои данные, система должна зарегистрировать пользователя
   3. Когда регистрация окончена, система должна отправить сообщение со ссылкой для активации аккаунта на почту пользователя

Пользователь:

1. Авторизация
   1. Когда пользователь вводит свои логин и пароль, система должна подтвердить логин пользователя
   2. Когда пользователь выбирает другой метод авторизации, система должна изменить метод авторизации на выбранный
   3. Когда пользователь вводит свой номер телефона, система должна отправить одноразовый СМС код на телефон пользователя для авторизации
2. Добавление пиццы в корзину
   1. Когда продукт со всеми параметрами выбирается, система должна сохранять заказ в корзине для пользователя
   2. Когда продукт со всеми параметрами удаляется, система должна удалить заказ из корзины
3. Ввод адреса
   1. Во время ввода персонального адреса, экран должен содержать поле для вводимых данных пользователем по адресу
   2. Во время ввода данных пользователем, система должна сохранять вводимые пользовательские данные
   3. Когда пользователь выбирает другой метод адрес, система должна сменить адрес на выбранный
4. Оплата заказа
   1. Когда происходит платеж картой онлайн, система должна обрабатывать платеж пользователя через банк
   2. Когда происходит платеж картой по терминалу, система должна обрабатывать платеж пользователя через банк
   3. Когда происходит платеж наличными, система должна создать чек на пиццу
   4. Когда происходит платеж переводом, система должна обрабатывать платеж пользователя через банк
5. Оформление заказа
   1. Когда проходит успешный платеж, система должна оформить заказ
   2. Когда происходит выбор пользователем платежа наличными или по терминалу, система должна оформить заказ
   3. Когда заказ оформлен, система должна создать заказ в базе данных заказов
6. Просмотр статуса заказа
   1. Система должна отображать статус заказа на экране, когда у заказа меняется статус
   2. Система должна получать статус заказ с пиццерии, когда у заказа меняется статус
   3. Система должна отправлять статус заказа в пиццерию, когда у заказа меняется статус

Оператор:

1. Добавление пиццы в корзину оператором
   1. Когда продукт со всеми параметрами выбирается, система должна сохранить заказ в корзине для пользователя
   2. Когда продукт со всеми параметрами отменяется, система должна удалить заказ из корзины
2. Принятие заказа по телефону оператором
   1. Когда пицца выбрана, система должна предоставить возможность выбрать топпинг на пиццу пользователю
   2. Когда проходит успешный платеж, система должна оформить заказ
   3. Когда происходит выбор пользователем платежа наличными или по терминалу, система должна оформить заказ
   4. Когда заказ оформлен, система должна создать заказ в базе данных заказов

Система:

1. Оповещение через СМС
   1. Система должна отправлять СМС с информацией о статусе заказе пользователю, когда у заказа меняется статус

Повар:

1. Готовка заказа поварами
   1. Когда заказ оформлен, система должна передать заказ на кухню для готовки
   2. Когда повар принимает заказ, система должна предоставить возможность сортировки заказов по номерам, времени принятия заказа в работу, сложности, размеру

Курьер:

1. Доставка заказа на дом курьером
   1. Когда заказ собран, система должна передать заказ на доставку курьеру
   2. Когда проходит платеж картой через терминал, система должна обработать платеж через банк
   3. Когда статус заказа меняется на доставлен, система должна закрыть заказ в базе данных заказов

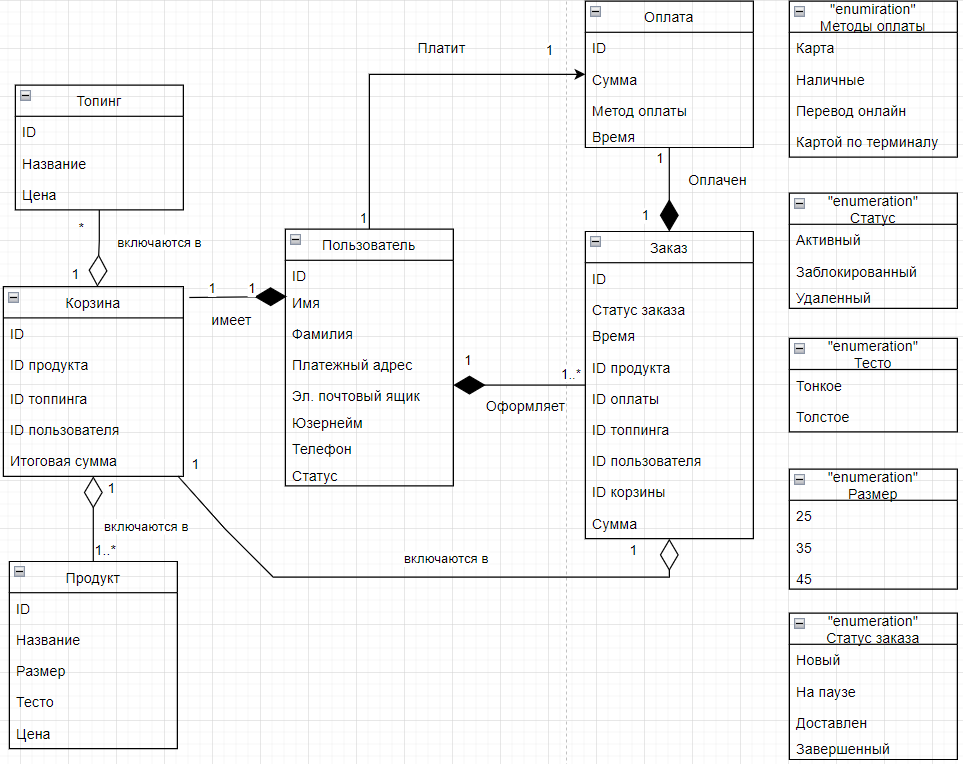
Ссылка на сценарии

https://skillfactroy.atlassian.net/wiki/spaces/TPS/pages/1114113

### **Логическая модель данных**

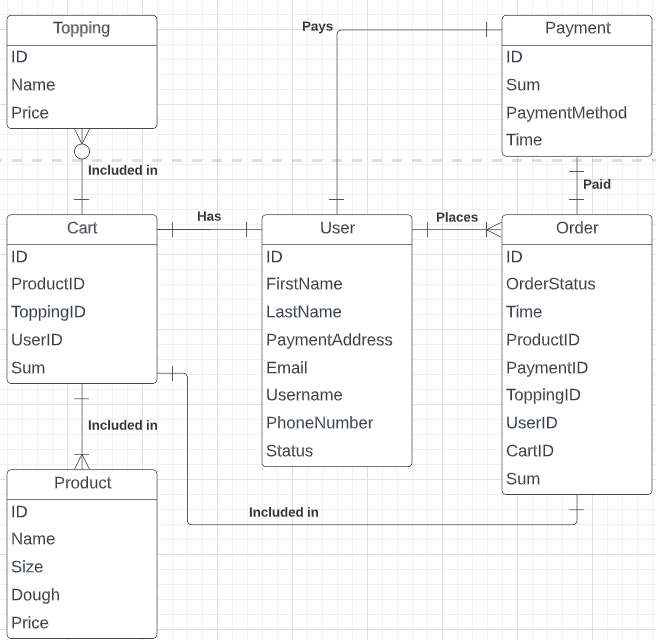
*[Включает описание основных сущностей, их атрибутов и связей — как минимум на концептуальном уровне.]*

[Если уже началось проектирование — то и логическую структуру БД. Здесь пригодятся UML Class Diagram или ERD.]



Ссылка на диаграмму:

<https://app.diagrams.net/#G1mQIlx-_8R7i9CHn3CW75rVYEg8IIQqgP>



Ссылка на диаграмму:

https://lucid.app/lucidchart/895ca5d5-e865-4805-ad4a-4e03b3f14430/edit?page=0\_0&invitationId=inv\_6ce4deb0-8beb-49b0-af3f-bb8f516ea7e6#

### **Программные интерфейсы**

*[Описание API, которой система будет предоставлять. Стоит расписать принципы, по которым строить API (REST/RPC/SOAP, файловый обмен и т. д.), методы, их входящие и возвращаемые параметры.]*

*[Можно ссылаться на объекты логической структуры данных. Если проектирование ещё не началось — можно описывать смысл методов, без всех подробностей.]*

* REST API:
  + Использование REST API вместо RPC и SOAP обуславливается относительной простотой создания, редактирования и использования программного интерфейса в системе.
  + Спецификация
    - OpenAPI
  + Инструмент
    - Swagger
      * Версия - 3.0.2
      * Основан на SmartBear
      * Документация Swagger:
        + https://swagger.io/docs/specification/about/
  + URL адрес - [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/)
  + API создано для работы с ресурсами пользователя, курьера, повара, оператора, корзины, товара, топпинга, заказа, остатка и платежей

#### Ресурс /user:

##### Метод GET ресурса /user:

* + - С помощью метода GET можно получить данные по пользователям, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{userId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| userId | integer | - | да | Идентификатор пользователя |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор пользователя |
| name | string | - | да | Имя пользователя |
| lastname | string | - | да | Фамилия пользователя |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик пользователя |
| password | string | - | да | Пароль пользователя |
| phone | integer | - | да | Номер телефона пользователя |
| payment\_address | string | - | да | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | да | Статус пользователя |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/user](https://api.mamamiapizzeria.com/data/2.5/v1/user)

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 888,

"name": "Anton",

"lastname": "Stepanov",

"username": "usop",

"email": "usop@gmail.com",

"password": "12345678",

"phone": +7 888 800 44 44,

"payment\_address": "Russia, Kazan, ul. Centr, d.1, kv.5",

"status": "Active"

},

{

"id": 777,

"name": "Sergey",

"lastname": "Momovich",

"username": "serg789",

"email": "movasd@google.com",

"password": "5555555",

"phone": +7 555 444 22 11,

"payment\_address": "Russia, Moscow, ul. Motornaya, d.12, kv.135",

"status": "Blocked"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/user?sort=asc](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 777,

"name": "Sergey",

"lastname": "Momovich",

"username": "serg789",

"email": "movasd@google.com",

"password": "5555555",

"phone": +7 555 444 22 11,

"payment\_address": "Russia, Moscow, ul. Motornaya, d.12, kv.135",

"status": "Blocked"

},

{

"id": 888,

"name": "Anton",

"lastname": "Stepanov",

"username": "usop",

"email": "usop@gmail.com",

"password": "12345678",

"phone": +7 888 800 44 44,

"payment\_address": "Russia, Kazan, ul. Centr, d.1, kv.5",

"status": "Active"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/user?filter[status]=active

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 888,

"name": "Anton",

"lastname": "Stepanov",

"username": "usop",

"email": "usop@gmail.com",

"password": "12345678",

"phone": +7 888 800 44 44,

"payment\_address": "Russia, Kazan, ul. Centr, d.1, kv.5",

"status": "Active"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/user](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)/777

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 777,

"name": "Sergey",

"lastname": "Momovich",

"username": "serg789",

"email": "movasd@google.com",

"password": "5555555",

"phone": +7 555 444 22 11,

"payment\_address": "Russia, Moscow, ul. Motornaya, d.12, kv.135",

"status": "Blocked"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

##### Метод GET ресурса /user/login:

* + - С помощью метода GET можно зарегистрировать сеанс пользователя в системе.
    - GET Login параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| username | string | - | да | Уникальный идентификатор пользователя |
| password | string | - | да | Пароль пользователя для входа в систему |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| code | integer | 200  400 | да | код ответа сервера |
| message | string | - | да | Сообщение о зарегистрированной сессии пользователя |

* + - Возвращаемые параметры header:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| X-RateLimit-Limit | integer | - | нет | Лимит запросов в час |
| X-RateLimit-Expires | string | - | нет | Дата по текущему времени Гринвича, когда истечет срок действия токена |

**Пример работы с ресурсом**

Пример GET параметра login

GET https://api.pizzeria.com/data/2.5/v1/user/login?username=username&password=password

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"code": 200,

"message": "logged in user session:1650806454274"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод GET ресурса /user/logout:

* + - С помощью метода GET можно завершить сеанс пользователя в системе
    - GET Logout параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Пример GET параметра logout

GET https://api.pizzeria.com/data/2.5/v1/user/logout

Ответ сервера:

HTTP 200 successful operation

##### Метод POST ресурса /user:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя пользователя |
| lastname | string | - | нет | Фамилия пользователя |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик пользователя |
| password | string | - | нет | Пароль пользователя |
| phone | integer | - | нет | Номер телефона пользователя |
| payment\_address | string | - | нет | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус пользователя |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор пользователя |
| name | string | - | да | Имя пользователя |
| lastname | string | - | да | Фамилия пользователя |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик пользователя |
| password | string | - | да | Пароль пользователя |
| phone | integer | - | да | Номер телефона пользователя |
| payment\_address | string | - | да | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | да | Статус пользователя |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/user](https://api.mamamiapizzeria.com/data/2.5/v1/user)

{

"name": "Roman",

"lastname": "Romanov",

"username": "Romanmon",

"email": "romanov@gmail.com",

"password": "123456",

"phone": +7 999 999 99 99,

"payment\_address": "Russia, Moscow, ul. Cheremushki, d.5, kv.13",

"status": "Active"

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 779

"name": "Roman",

"lastname": "Romanov",

"username": "Romanmon",

"email": "romanov@gmail.com",

"password": "123456",

"phone": +7 999 999 99 99,

"payment\_address": "Russia, Moscow, ul. Cheremushki, d.5, kv.13",

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /user/{userId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{userId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| userId | integer | - | да | Идентификатор пользователя |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя пользователя |
| lastname | string | - | нет | Фамилия пользователя |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик пользователя |
| password | string | - | нет | Пароль пользователя |
| phone | integer | - | нет | Номер телефона пользователя |
| payment\_address | string | - | нет | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус пользователя |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор пользователя |
| name | string | - | да | Имя пользователя |
| lastname | string | - | да | Фамилия пользователя |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик пользователя |
| password | string | - | да | Пароль пользователя |
| phone | integer | - | да | Номер телефона пользователя |
| payment\_address | string | - | да | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | да | Статус пользователя |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/user](https://api.mamamiapizzeria.com/data/2.5/v1/user)/777

{

"name": "Shier",

"lastname": "Guzanb",

"username": "Guze",

"email": "guzanb@gmail.com",

"password": "098765",

"phone": +7 977 925 88 99,

"payment\_address": "Russia, Moscow, ul. Krasnaya, d.5, kv.13",

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 777

"name": "Shier",

"lastname": "Guzanb",

"username": "Guze",

"email": "guzanb@gmail.com",

"password": "098765",

"phone": +7 977 925 88 99,

"payment\_address": "Russia, Moscow, ul. Krasnaya, d.5, kv.13",

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

##### Метод PATCH ресурса /user/{userId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{userId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| userId | integer | - | да | Идентификатор пользователя |

* + - PATCH Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя пользователя |
| lastname | string | - | нет | Фамилия пользователя |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик пользователя |
| password | string | - | нет | Пароль пользователя |
| phone | integer | - | нет | Номер телефона пользователя |
| payment\_address | string | - | нет | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус пользователя |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор пользователя |
| name | string | - | да | Имя пользователя |
| lastname | string | - | да | Фамилия пользователя |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик пользователя |
| password | string | - | да | Пароль пользователя |
| phone | integer | - | да | Номер телефона пользователя |
| payment\_address | string | - | да | Адрес платежа пользователя |
| status | string | Активный  Заблокированный  Удаленный | да | Статус пользователя |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/user](https://api.mamamiapizzeria.com/data/2.5/v1/user)/777

{

"name": "Roman",

"lastname": "Romanov",

"username": "Romanmon",

"email": "roma@gmail.com",

"password": "4444444",

"phone": +7 999 999 99 11,

"payment\_address": "Russia, Moscow, ul. Cheremushki, d.5, kv.13",

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 777

"name": "Roman",

"lastname": "Romanov",

"username": "Romanmon",

"email": "roma@gmail.com",

"password": "4444444",

"phone": +7 999 999 99 11,

"payment\_address": "Russia, Moscow, ul. Cheremushki, d.5, kv.13",

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

##### Метод DELETE ресурса /user/{userId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{userId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| userId | integer | - | да | Идентификатор пользователя |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/user/777

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 user not found

Content-Type: application/json

{

"code": 404,

"message": "user not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /deliveryman:

##### Метод GET ресурса /deliveryman:

* + - С помощью метода GET можно получить данные по курьерам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{deliverymanId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| deliverymanId | integer | - | да | Идентификатор курьера |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор курьера |
| name | string | - | да | Имя курьера |
| lastname | string | - | да | Фамилия курьера |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик курьера |
| password | string | - | да | Пароль курьера |
| phone | integer | - | да | Номер телефона курьера |
| number\_of\_deliveries | integer | - | да | Кол-во доставок |
| salary | integer | - | да | Зарплата курьера |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус курьера |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)deliveryman

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 555,

"name": "Kupel",

"lastname": "Asud",

"username": "asd",

"email": "asd@gmail.com",

"password": "12345678",

"phone": +7 444 800 44 44,

"number\_of\_deliveries": 50,

"salary": 80000,

"order\_id": 123,

"status": "Active"

},

{

"id": 556,

"name": "Sergey",

"lastname": "Petrovich",

"username": "petr",

"email": "serp@google.com",

"password": "5555555",

"phone": +7 444 222 22 22,

"number\_of\_deliveries": 21,

"salary": 0,

"order\_id": 124,

"status": "Blocked"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/deliveryman?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)desc

Ответ сервера:

[

{

"id": 556,

"name": "Sergey",

"lastname": "Petrovich",

"username": "petr",

"email": "serp@google.com",

"password": "5555555",

"phone": +7 444 222 22 22,

"number\_of\_deliveries": 21,

"salary": 0,

"order\_id": 123,

"status": "Blocked"

},

{

"id": 555,

"name": "Kupel",

"lastname": "Asud",

"username": "asd",

"email": "asd@gmail.com",

"password": "12345678",

"phone": +7 444 800 44 44,

"number\_of\_deliveries": 50,

"salary": 80000,

"order\_id": 124,

"status": "Active"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/deliveryman?filter[status]=blocked

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 556,

"name": "Sergey",

"lastname": "Petrovich",

"username": "petr",

"email": "serp@google.com",

"password": "5555555",

"phone": +7 444 222 22 22,

"number\_of\_deliveries": 21,

"salary": 0,

"order\_id": 123,

"status": "Blocked"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)deliveryman/556

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 556,

"name": "Sergey",

"lastname": "Petrovich",

"username": "petr",

"email": "serp@google.com",

"password": "5555555",

"phone": +7 444 222 22 22,

"number\_of\_deliveries": 21,

"salary": 0,

"order\_id": 123,

"status": "Blocked"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

* + Метод GET ресурса /deliveryman/login:
    - С помощью метода GET можно зарегистрировать сеанс курьера в системе.
    - GET Login параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| username | string | - | да | Уникальный идентификатор пользователя |
| password | string | - | да | Пароль пользователя для входа в систему |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| code | integer | 200  400 | да | код ответа сервера |
| message | string | - | да | Сообщение о зарегистрированной сессии пользователя |

* + - Возвращаемые параметры header:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| X-RateLimit-Limit | integer | - | нет | Лимит запросов в час |
| X-RateLimit-Expires | string | - | нет | Дата по текущему времени Гринвича, когда истечет срок действия токена |

**Пример работы с ресурсом**

Пример GET параметра login

GET https://api.pizzeria.com/data/2.5/v1/deliveryman/login?username=username&password=password

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"code": 200,

"message": "logged in user session:1650806454285"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

* + Метод GET ресурса /deliveryman/logout:
    - С помощью метода GET можно завершить сеанс курьера в системе
    - GET Logout параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Пример GET параметра logout

GET https://api.pizzeria.com/data/2.5/v1/deliveryman/logout

Ответ сервера:

HTTP 200 successful operation

##### Метод POST ресурса /deliveryman:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя курьера |
| lastname | string | - | нет | Фамилия курьера |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик курьера |
| password | string | - | нет | Пароль курьера |
| phone | integer | - | нет | Номер телефона курьера |
| number\_of\_deliveries | integer | - | нет | Кол-во доставок |
| salary | integer | - | нет | Зарплата курьера |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус курьера |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор курьера |
| name | string | - | да | Имя курьера |
| lastname | string | - | да | Фамилия курьера |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик курьера |
| password | string | - | да | Пароль курьера |
| phone | integer | - | да | Номер телефона курьера |
| number\_of\_deliveries | integer | - | да | Кол-во доставок |
| salary | integer | - | да | Зарплата курьера |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус курьера |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)deliveryman

{

"name": "Alex",

"lastname": "Deltovich",

"username": "Delta",

"email": "alexp@google.com",

"password": "alex46697",

"phone": +7 111 222 22 22,

"number\_of\_deliveries": 0,

"salary": 60000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 557

"name": "Alex",

"lastname": "Deltovich",

"username": "Delta",

"email": "alexp@google.com",

"password": "alex46697",

"phone": +7 111 222 22 22,

"number\_of\_deliveries": 0,

"salary": 60000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /deliveryman/{deliverymanId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{deliverymanId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| deliverymanId | integer | - | да | Идентификатор курьера |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя курьера |
| lastname | string | - | нет | Фамилия курьера |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик курьера |
| password | string | - | нет | Пароль курьера |
| phone | integer | - | нет | Номер телефона курьера |
| number\_of\_deliveries | integer | - | нет | Кол-во доставок |
| salary | integer | - | нет | Зарплата курьера |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус курьера |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор курьера |
| name | string | - | да | Имя курьера |
| lastname | string | - | да | Фамилия курьера |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик курьера |
| password | string | - | да | Пароль курьера |
| phone | integer | - | да | Номер телефона курьера |
| number\_of\_deliveries | integer | - | да | Кол-во доставок |
| salary | integer | - | да | Зарплата курьера |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус курьера |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)deliveryman/555

{

"name": "Json",

"lastname": "Xml",

"username": "API",

"email": "openapi@gmail.com",

"password": "password",

"phone": +7 444 800 44 44,

"number\_of\_deliveries": 0,

"salary": 0,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 555

"name": "Json",

"lastname": "Xml",

"username": "API",

"email": "openapi@gmail.com",

"password": "password",

"phone": +7 444 800 44 44,

"number\_of\_deliveries": 0,

"salary": 0,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

##### Метод PATCH ресурса /deliveryman/{deliverymanId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{deliverymanId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| deliverymanId | integer | - | да | Идентификатор курьера |

* + - PATCH Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя курьера |
| lastname | string | - | нет | Фамилия курьера |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик курьера |
| password | string | - | нет | Пароль курьера |
| phone | integer | - | нет | Номер телефона курьера |
| number\_of\_deliveries | integer | - | нет | Кол-во доставок |
| salary | integer | - | нет | Зарплата курьера |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус курьера |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор курьера |
| name | string | - | да | Имя курьера |
| lastname | string | - | да | Фамилия курьера |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик курьера |
| password | string | - | да | Пароль курьера |
| phone | integer | - | да | Номер телефона курьера |
| number\_of\_deliveries | integer | - | да | Кол-во доставок |
| salary | integer | - | да | Зарплата курьера |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус курьера |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)deliveryman/555

{

"name": "Kupel",

"lastname": "Asud",

"username": "asd",

"email": "reserve@gmail.com",

"password": "verydifficultpassword",

"phone": +7 444 800 44 44,

"number\_of\_deliveries": 50,

"salary": 80000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 555

"name": "Kupel",

"lastname": "Asud",

"username": "asd",

"email": "reserve@gmail.com",

"password": "verydifficultpassword",

"phone": +7 444 800 44 44,

"number\_of\_deliveries": 50,

"salary": 80000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

##### Метод DELETE ресурса /deliveryman/{deliverymanId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{deliverymanId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| deliverymanId | integer | - | да | Идентификатор курьера |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/deliveryman/555

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 deliveryman not found

Content-Type: application/json

{

"code": 404,

"message": "deliveryman not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /chef:

##### Метод GET ресурса /chef:

* + - С помощью метода GET можно получить данные по поварам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{chefId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| chefId | integer | - | да | Идентификатор повара |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор повара |
| name | string | - | да | Имя повара |
| lastname | string | - | да | Фамилия повара |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик повара |
| password | string | - | да | Пароль повара |
| phone | integer | - | да | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | да | Кол-во приготовленных пицц |
| salary | integer | - | да | Зарплата повара |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус повара |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)chef

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 333,

"name": "Asura",

"lastname": "Loki",

"username": "qwe",

"email": "qwe@gmail.com",

"password": "1234567\*",

"phone": +7 445 800 44 44,

"number\_of\_meals\_prepared": 10,

"salary": 100000,

"order\_id": 123,

"status": "Active"

},

{

"id": 334,

"name": "Aki",

"lastname": "Takiyaba",

"username": "akicha",

"email": "numer@google.com",

"password": "5555555!",

"phone": +7 444 222 50 77,

"number\_of\_meals\_prepared": 40,

"salary": 0,

"order\_id": 123,

"status": "Terminated"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/chef?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 333,

"name": "Asura",

"lastname": "Loki",

"username": "qwe",

"email": "qwe@gmail.com",

"password": "1234567\*",

"phone": +7 445 800 44 44,

"number\_of\_meals\_prepared": 10,

"salary": 100000,

"order\_id": 123,

"status": "Active"

},

{

"id": 334,

"name": "Aki",

"lastname": "Takiyaba",

"username": "akicha",

"email": "numer@google.com",

"password": "5555555!",

"phone": +7 444 222 50 77,

"number\_of\_meals\_prepared": 40,

"salary": 0,

"order\_id": 123,

"status": "Terminated"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/chef?filter[status]=blocked

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 385,

"name": "Sergey",

"lastname": "Ser",

"username": "setr",

"email": "perp@google.com",

"password": "5555551",

"phone": +7 448 222 22 22,

"number\_of\_meals\_prepared": 5,

"salary": 0,

"order\_id": 123,

"status": "Blocked"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)chef/333

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 333,

"name": "Asura",

"lastname": "Loki",

"username": "qwe",

"email": "qwe@gmail.com",

"password": "1234567\*",

"phone": +7 445 800 44 44,

"number\_of\_meals\_prepared": 10,

"salary": 100000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

* + Метод GET ресурса /chef/login:
    - С помощью метода GET можно зарегистрировать сеанс повара в системе.
    - GET Login параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| username | string | - | да | Уникальный идентификатор пользователя |
| password | string | - | да | Пароль пользователя для входа в систему |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| code | integer | 200  400 | да | код ответа сервера |
| message | string | - | да | Сообщение о зарегистрированной сессии пользователя |

* + - Возвращаемые параметры header:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| X-RateLimit-Limit | integer | - | нет | Лимит запросов в час |
| X-RateLimit-Expires | string | - | нет | Дата по текущему времени Гринвича, когда истечет срок действия токена |

**Пример работы с ресурсом**

Пример GET параметра login

GET https://api.pizzeria.com/data/2.5/v1/chef/login?username=username&password=password

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"code": 200,

"message": "logged in user session:1650806454285"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

* + Метод GET ресурса /chef/logout:
    - С помощью метода GET можно завершить сеанс повара в системе
    - GET Logout параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Пример GET параметра logout

GET https://api.pizzeria.com/data/2.5/v1/chef/logout

Ответ сервера:

HTTP 200 successful operation

##### Метод POST ресурса /chef:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя повара |
| lastname | string | - | нет | Фамилия повара |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик повара |
| password | string | - | нет | Пароль повара |
| phone | integer | - | нет | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | нет | Кол-во приготовленных пицц |
| salary | integer | - | нет | Зарплата повара |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус повара |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор повара |
| name | string | - | да | Имя повара |
| lastname | string | - | да | Фамилия повара |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик повара |
| password | string | - | да | Пароль повара |
| phone | integer | - | да | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | да | Кол-во приготовленных пицц |
| salary | integer | - | да | Зарплата повара |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус повара |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)chef

{

"name": "L",

"lastname": "B",

"username": "A",

"email": "cp@google.com",

"password": "lbac",

"phone": +7 302 222 22 22,

"number\_of\_meals\_prepared": 0,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 338

"name": "L",

"lastname": "B",

"username": "A",

"email": "cp@google.com",

"password": "lbac",

"phone": +7 302 222 22 22,

"number\_of\_meals\_prepared": 0,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /chef/{chefId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{chefId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| chefId | integer | - | да | Идентификатор повара |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя повара |
| lastname | string | - | нет | Фамилия повара |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик повара |
| password | string | - | нет | Пароль повара |
| phone | integer | - | нет | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | нет | Кол-во приготовленных пицц |
| salary | integer | - | нет | Зарплата повара |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус повара |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор повара |
| name | string | - | да | Имя повара |
| lastname | string | - | да | Фамилия повара |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик повара |
| password | string | - | да | Пароль повара |
| phone | integer | - | да | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | да | Кол-во приготовленных пицц |
| salary | integer | - | да | Зарплата повара |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус повара |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)chef/333

{

"name": "Asd",

"lastname": "Asd",

"username": "Asd",

"email": "Asd@gmail.com",

"password": "1",

"phone": +7 445 100 44 44,

"number\_of\_meals\_prepared": 0,

"salary": 500000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 333,

"name": "Asd",

"lastname": "Asd",

"username": "Asd",

"email": "Asd@gmail.com",

"password": "1",

"phone": +7 445 100 44 44,

"number\_of\_meals\_prepared": 0,

"salary": 500000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

##### Метод PATCH ресурса /chef/{chefId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{chefId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| chefId | integer | - | да | Идентификатор повара |

* + - PATCH Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя повара |
| lastname | string | - | нет | Фамилия повара |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик повара |
| password | string | - | нет | Пароль повара |
| phone | integer | - | нет | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | нет | Кол-во приготовленных пицц |
| salary | integer | - | нет | Зарплата повара |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус повара |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор повара |
| name | string | - | да | Имя повара |
| lastname | string | - | да | Фамилия повара |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик повара |
| password | string | - | да | Пароль повара |
| phone | integer | - | да | Номер телефона повара |
| number\_of\_meals\_prepared | integer | - | да | Кол-во приготовленных пицц |
| salary | integer | - | да | Зарплата повара |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус повара |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)chef/333

{

"name": "Asura",

"lastname": "LOLICH",

"username": "qwe",

"email": "qwe@gmail.com",

"password": "LOL",

"phone": +7 445 800 44 44,

"number\_of\_meals\_prepared": 10,

"salary": 100000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 333,

"name": "Asura",

"lastname": "LOLICH",

"username": "qwe",

"email": "qwe@gmail.com",

"password": "LOL",

"phone": +7 445 800 44 44,

"number\_of\_meals\_prepared": 10,

"salary": 100000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

##### Метод DELETE ресурса /chef/{chefId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{chefId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| chefId | integer | - | да | Идентификатор повара |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/chef/333

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 chef not found

Content-Type: application/json

{

"code": 404,

"message": "chef not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /operator:

##### Метод GET ресурса /operator:

* + - С помощью метода GET можно получить данные по поварам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{operatorId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| operatorId | integer | - | да | Идентификатор повара |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор оператора |
| name | string | - | да | Имя оператора |
| lastname | string | - | да | Фамилия оператора |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик оператора |
| password | string | - | да | Пароль оператора |
| phone | integer | - | да | Номер телефона оператора |
| number\_of\_calls | integer | - | да | Кол-во звонков |
| salary | integer | - | да | Зарплата оператора |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус оператора |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)operator

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 789,

"name": "Lili",

"lastname": "Chan",

"username": "qwa",

"email": "kwerk@gmail.com",

"password": "nihongodayo",

"phone": +7 789 800 44 44,

"number\_of\_calls": 1000,

"salary": 150000,

"order\_id": 123,

"status": "Active"

},

{

"id": 850,

"name": "Tori",

"lastname": "Takeda",

"username": "chan",

"email": "torichan@google.com",

"password": "456872;asd",

"phone": +7 444 349 50 77,

"number\_of\_calls": 40,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/operator?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 789,

"name": "Lili",

"lastname": "Chan",

"username": "qwa",

"email": "kwerk@gmail.com",

"password": "nihongodayo",

"phone": +7 789 800 44 44,

"number\_of\_calls": 1000,

"salary": 150000,

"order\_id": 123,

"status": "Active"

},

{

"id": 850,

"name": "Tori",

"lastname": "Takeda",

"username": "chan",

"email": "torichan@google.com",

"password": "456872;asd",

"phone": +7 444 349 50 77,

"number\_of\_calls": 40,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/operator?filter[status]=active

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 789,

"name": "Lili",

"lastname": "Chan",

"username": "qwa",

"email": "kwerk@gmail.com",

"password": "nihongodayo",

"phone": +7 789 800 44 44,

"number\_of\_calls": 1000,

"salary": 150000,

"order\_id": 123,

"status": "Active"

},

{

"id": 850,

"name": "Tori",

"lastname": "Takeda",

"username": "chan",

"email": "torichan@google.com",

"password": "456872;asd",

"phone": +7 444 349 50 77,

"number\_of\_calls": 40,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)operator/850

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 850,

"name": "Tori",

"lastname": "Takeda",

"username": "chan",

"email": "torichan@google.com",

"password": "456872;asd",

"phone": +7 444 349 50 77,

"number\_of\_calls": 40,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

* + Метод GET ресурса /operator/login:
    - С помощью метода GET можно зарегистрировать сеанс оператора в системе.
    - GET Login параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| username | string | - | да | Уникальный идентификатор пользователя |
| password | string | - | да | Пароль пользователя для входа в систему |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| code | integer | 200  400 | да | код ответа сервера |
| message | string | - | да | Сообщение о зарегистрированной сессии пользователя |

* + - Возвращаемые параметры header:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| X-RateLimit-Limit | integer | - | нет | Лимит запросов в час |
| X-RateLimit-Expires | string | - | нет | Дата по текущему времени Гринвича, когда истечет срок действия токена |

**Пример работы с ресурсом**

Пример GET параметра login

GET https://api.pizzeria.com/data/2.5/v1/operator/login?username=username&password=password

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"code": 200,

"message": "logged in user session:1650806454285"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

* + Метод GET ресурса /operator/logout:
    - С помощью метода GET можно завершить сеанс оператора в системе
    - GET Logout параметры:

| Название поля | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Пример GET параметра logout

GET https://api.pizzeria.com/data/2.5/v1/operator/logout

Ответ сервера:

HTTP 200 successful operation

##### Метод POST ресурса /operator:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя оператора |
| lastname | string | - | нет | Фамилия оператора |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик оператора |
| password | string | - | нет | Пароль оператора |
| phone | integer | - | нет | Номер телефона оператора |
| number\_of\_calls | integer | - | нет | Кол-во звонков |
| salary | integer | - | нет | Зарплата оператора |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус оператора |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор оператора |
| name | string | - | да | Имя оператора |
| lastname | string | - | да | Фамилия оператора |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик оператора |
| password | string | - | да | Пароль оператора |
| phone | integer | - | да | Номер телефона оператора |
| number\_of\_calls | integer | - | да | Кол-во звонков |
| salary | integer | - | да | Зарплата оператора |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус оператора |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)operator

{

"name": "Abe",

"lastname": "Chiba",

"username": "Loki",

"email": "ab@google.com",

"password": "chibaabelokisan1",

"phone": +7 517 222 22 22,

"number\_of\_calls": 0,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 799

"name": "Abe",

"lastname": "Chiba",

"username": "Loki",

"email": "ab@google.com",

"password": "chibaabelokisan1",

"phone": +7 517 222 22 22,

"number\_of\_calls": 0,

"salary": 90000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /operator/{operatorId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{operatorId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| operatorId | integer | - | да | Идентификатор оператора |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя оператора |
| lastname | string | - | нет | Фамилия оператора |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик оператора |
| password | string | - | нет | Пароль оператора |
| phone | integer | - | нет | Номер телефона оператора |
| number\_of\_calls | integer | - | нет | Кол-во звонков |
| salary | integer | - | нет | Зарплата оператора |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус оператора |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор оператора |
| name | string | - | да | Имя оператора |
| lastname | string | - | да | Фамилия оператора |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик оператора |
| password | string | - | да | Пароль оператора |
| phone | integer | - | да | Номер телефона оператора |
| number\_of\_calls | integer | - | да | Кол-во звонков |
| salary | integer | - | да | Зарплата оператора |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус оператора |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)operator/850

{

"name": "Kebab",

"lastname": "From Turkey",

"username": "Super\_Kebab",

"email": "kebab@google.com",

"password": "kebabrules",

"phone": +7 963 349 50 77,

"number\_of\_calls": 0,

"salary": 60000,

"order\_id": 123,

"status": "Active"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 850,

"name": "Kebab",

"lastname": "From Turkey",

"username": "Super\_Kebab",

"email": "kebab@google.com",

"password": "kebabrules",

"phone": +7 963 349 50 77,

"number\_of\_calls": 0,

"salary": 60000,

"order\_id": 123,

"status": "Active"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

##### Метод PATCH ресурса /operator/{operatorId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{operatorId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| operatorId | integer | - | да | Идентификатор оператора |

* + - PATCH Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Имя оператора |
| lastname | string | - | нет | Фамилия оператора |
| username | string | - | нет | Уникальный идентификатор пользователя для логина |
| email | string | - | нет | Электронный ящик оператора |
| password | string | - | нет | Пароль оператора |
| phone | integer | - | нет | Номер телефона оператора |
| number\_of\_calls | integer | - | нет | Кол-во звонков |
| salary | integer | - | нет | Зарплата оператора |
| order\_id | integer | - | нет | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | нет | Статус оператора |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор оператора |
| name | string | - | да | Имя оператора |
| lastname | string | - | да | Фамилия оператора |
| username | string | - | да | Уникальный идентификатор пользователя для логина |
| email | string | - | да | Электронный ящик оператора |
| password | string | - | да | Пароль оператора |
| phone | integer | - | да | Номер телефона оператора |
| number\_of\_calls | integer | - | да | Кол-во звонков |
| salary | integer | - | да | Зарплата оператора |
| order\_id | integer | - | да | Идентификатор заказа |
| status | string | Активный  Заблокированный  Удаленный | да | Статус оператора |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)operator/850

{

"name": "Tori",

"lastname": "Takeda",

"username": "Sugoi\_chan",

"email": "Sugo-tekkain@google.com",

"password": "456872;asd",

"phone": +7 444 349 50 77,

"number\_of\_calls": 40,

"salary": 90000,

"order\_id": 123,

"status": "Blocked"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 850,

"name": "Tori",

"lastname": "Takeda",

"username": "Sugoi\_chan",

"email": "Sugo-tekkain@google.com",

"password": "456872;asd",

"phone": +7 444 349 50 77,

"number\_of\_calls": 40,

"salary": 90000,

"order\_id": 123,

"status": "Blocked"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

##### Метод DELETE ресурса /operator/{operatorId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{operatorId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| operatorId | integer | - | да | Идентификатор оператора |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/operator/850

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 operator not found

Content-Type: application/json

{

"code": 404,

"message": "operator not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /cart:

##### Метод GET ресурса /cart:

* + - С помощью метода GET можно получить данные по корзинам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{cartId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| cartId | integer | - | да | Идентификатор корзины |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| topping\_id | integer | - | да | Идентификатор топпинга |
| user\_id | integer | - | да | Идентификатор пользователя |
| total\_price | integer | - | да | Итоговая сумма продукта и топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)cart

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 111,

"product\_id": 999,

"topping\_id": 222,

"user\_id": 777,

"total\_price": 100

},

{

"id": 112,

"product\_id": 998,

"topping\_id": 223,

"user\_id": 778,

"total\_price": 120

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/cart?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 111,

"product\_id": 999,

"topping\_id": 222,

"user\_id": 777,

"total\_price": 100

},

{

"id": 112,

"product\_id": 998,

"topping\_id": 223,

"user\_id": 778,

"total\_price": 120

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/cart?filter[total\_price]=100

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 111,

"product\_id": 999,

"topping\_id": 222,

"user\_id": 777,

"total\_price": 100

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)cart/111

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 111,

"product\_id": 999,

"topping\_id": 222,

"user\_id": 777,

"total\_price": 100

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

##### Метод POST ресурса /cart:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| product\_id | integer | - | нет | Идентификатор продукта |
| topping\_id | integer | - | нет | Идентификатор топпинга |
| user\_id | integer | - | нет | Идентификатор пользователя |
| total\_price | integer | - | нет | Итоговая сумма продукта и топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| topping\_id | integer | - | да | Идентификатор топпинга |
| user\_id | integer | - | да | Идентификатор пользователя |
| total\_price | integer | - | да | Итоговая сумма продукта и топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)cart

{

"product\_id": 1,

"topping\_id": 2,

"user\_id": 3,

"total\_price": 50

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 110,

"product\_id": 1,

"topping\_id": 2,

"user\_id": 3,

"total\_price": 50

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /cart/{cartId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{cartId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| cartId | integer | - | да | Идентификатор корзины |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| product\_id | integer | - | нет | Идентификатор продукта |
| topping\_id | integer | - | нет | Идентификатор топпинга |
| user\_id | integer | - | нет | Идентификатор пользователя |
| total\_price | integer | - | нет | Итоговая сумма продукта и топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| topping\_id | integer | - | да | Идентификатор топпинга |
| user\_id | integer | - | да | Идентификатор пользователя |
| total\_price | integer | - | да | Итоговая сумма продукта и топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)cart/111

{

"product\_id": 10,

"topping\_id": 11,

"user\_id": 12,

"total\_price": 100

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 111,

"product\_id": 10,

"topping\_id": 11,

"user\_id": 12,

"total\_price": 100

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

##### Метод PATCH ресурса /cart/{cartId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{cartId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| cartId | integer | - | да | Идентификатор корзины |

* + - PATCH Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| product\_id | integer | - | нет | Идентификатор продукта |
| topping\_id | integer | - | нет | Идентификатор топпинга |
| user\_id | integer | - | нет | Идентификатор пользователя |
| total\_price | integer | - | нет | Итоговая сумма продукта и топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| topping\_id | integer | - | да | Идентификатор топпинга |
| user\_id | integer | - | да | Идентификатор пользователя |
| total\_price | integer | - | да | Итоговая сумма продукта и топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)cart/111

{

"product\_id": 5,

"topping\_id": 6,

"user\_id": 777,

"total\_price": 100

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 111,

"product\_id": 5,

"topping\_id": 6,

"user\_id": 777,

"total\_price": 100

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

##### Метод DELETE ресурса /cart/{cartId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{cartId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| cartId | integer | - | да | Идентификатор корзины |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/cart/111

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 cart not found

Content-Type: application/json

{

"code": 404,

"message": "cart not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /product:

##### Метод GET ресурса /product:

* + - С помощью метода GET можно получить данные по корзинам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{productId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| productId | integer | - | да | Идентификатор продукта |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор продукта |
| name | string | - | да | Название продукта |
| size | integer | 25  35  45 | да | Размер продукта |
| dough | string | тонкое  толстое | да | Тесто продукта |
| price | integer | - | да | Цена продукта |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)product

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 999,

"name": "cheese",

"size": 25,

"dough": "тонкое",

"price": 400

},

{

"id": 1000,

"name": "pepperoni",

"size": 35,

"dough": "толстое",

"price": 600

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/product?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 999,

"name": "cheese",

"size": 25,

"dough": "тонкое",

"price": 400

},

{

"id": 1000,

"name": "pepperoni",

"size": 35,

"dough": "толстое",

"price": 600

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/product?filter[dough]=тонкое

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 999,

"name": "cheese",

"size": 25,

"dough": "тонкое",

"price": 400

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)product/1000

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 1000,

"name": "pepperoni",

"size": 35,

"dough": "толстое",

"price": 600

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

##### Метод POST ресурса /product:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Название продукта |
| size | integer | 25  35  45 | нет | Размер продукта |
| dough | string | тонкое  толстое | нет | Тесто продукта |
| price | integer | - | нет | Цена продукта |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор продукта |
| name | string | - | да | Название продукта |
| size | integer | 25  35  45 | да | Размер продукта |
| dough | string | тонкое  толстое | да | Тесто продукта |
| price | integer | - | да | Цена продукта |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)product

{

"name": "bacon",

"size": 25,

"dough": "толстое",

"price": 500

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 1001,

"name": "bacon",

"size": 25,

"dough": "толстое",

"price": 500

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /product/{productId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{productId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| productId | integer | - | да | Идентификатор продукта |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Название продукта |
| size | integer | 25  35  45 | нет | Размер продукта |
| dough | string | тонкое  толстое | нет | Тесто продукта |
| price | integer | - | нет | Цена продукта |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор продукта |
| name | string | - | да | Название продукта |
| size | integer | 25  35  45 | да | Размер продукта |
| dough | string | тонкое  толстое | да | Тесто продукта |
| price | integer | - | да | Цена продукта |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)product/1000

{

"name": "super",

"size": 45,

"dough": "тонкое",

"price": 800

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 1000,

"name": "super",

"size": 45,

"dough": "тонкое",

"price": 800

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

##### Метод PATCH ресурса /product/{productId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{productId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| productId | integer | - | да | Идентификатор продукта |

* + - PATCH Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Название продукта |
| size | integer | 25  35  45 | нет | Размер продукта |
| dough | string | тонкое  толстое | нет | Тесто продукта |
| price | integer | - | нет | Цена продукта |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор продукта |
| name | string | - | да | Название продукта |
| size | integer | 25  35  45 | да | Размер продукта |
| dough | string | тонкое  толстое | да | Тесто продукта |
| price | integer | - | да | Цена продукта |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)product/1000

{

"name": "pepperoni",

"size": 45,

"dough": "толстое",

"price": 600

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 1000,

"name": "pepperoni",

"size": 45,

"dough": "толстое",

"price": 600

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

##### Метод DELETE ресурса /product/{productId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{productId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| productId | integer | - | да | Идентификатор продукта |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/product/1000

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 product not found

Content-Type: application/json

{

"code": 404,

"message": "product not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /topping:

##### Метод GET ресурса /topping:

* + - С помощью метода GET можно получить данные по корзинам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{toppingId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| toppingId | integer | - | да | Идентификатор топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор топпинга |
| name | string | - | да | Название топпинга |
| price | integer | - | да | Цена топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)topping

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 222,

"name": "cheese",

"price": 50

},

{

"id": 223,

"name": "sause",

"price": 40

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 topping not found

Content-Type: application/json

{

"code": 404,

"message": "topping not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/topping?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 222,

"name": "cheese",

"price": 50

},

{

"id": 223,

"name": "sause",

"price": 40

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 topping not found

Content-Type: application/json

{

"code": 404,

"message": "topping not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/topping?filter[name]=cheese

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 222,

"name": "cheese",

"price": 50

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 topping not found

Content-Type: application/json

{

"code": 404,

"message": "topping not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)topping/222

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 222,

"name": "cheese",

"price": 50

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 topping not found

Content-Type: application/json

{

"code": 404,

"message": "topping not found"

}

##### Метод POST ресурса /topping:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Название топпинга |
| price | integer | - | нет | Цена топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор топпинга |
| name | string | - | да | Название топпинга |
| price | integer | - | да | Цена топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)topping

{

"name": "umai",

"price": 60

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 225,

"name": "umai",

"price": 60

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /topping/{toppingId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{toppingId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| toppingId | integer | - | да | Идентификатор топпинга |

* + - PUT Параметры тела:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | нет | Название топпинга |
| price | integer | - | нет | Цена топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор топпинга |
| name | string | - | да | Название топпинга |
| price | integer | - | да | Цена топпинга |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)topping/222

{

"name": "cheese",

"price": 50

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 222,

"name": "cheese",

"price": 50

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 topping not found

Content-Type: application/json

{

"code": 404,

"message": "topping not found"

}

##### Метод DELETE ресурса /topping/{toppingId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{toppingId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| toppingId | integer | - | да | Идентификатор топпинга |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/topping/222

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 topping not found

Content-Type: application/json

{

"code": 404,

"message": "topping not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /order:

##### Метод GET ресурса /order:

* + - С помощью метода GET можно получить данные по корзинам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{orderId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| orderId | integer | - | да | Идентификатор заказа |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор заказа |
| user\_id | integer | - | да | Идентификатор пользователя |
| cart\_id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| payment\_id | integer | - | да | Идентификатор оплаты |
| topping\_id | integer | - | да | Идентификатор топпинга |
| time | string | - | да | Время заказа |
| price | integer | - | да | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | да | Статус заказа |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)order

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 123,

"user\_id": 777,

"cart\_id": 111,

"product\_id": 999,

"payment\_id": 456,

"topping\_id": 222,

"time": "2022-01-26T10:51:13.023Z",

"price": 450,

"status": "New"

},

{

"id": 124,

"user\_id": 778,

"cart\_id": 112,

"product\_id": 1000,

"payment\_id": 457,

"topping\_id": 223,

"time": "2022-04-26T10:51:13.023Z",

"price": 600,

"status": "Delivered"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/order?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 123,

"user\_id": 777,

"cart\_id": 111,

"product\_id": 999,

"payment\_id": 456,

"topping\_id": 222,

"time": "2022-01-26T10:51:13.023Z",

"price": 450,

"status": "New"

},

{

"id": 124,

"user\_id": 778,

"cart\_id": 112,

"product\_id": 1000,

"payment\_id": 457,

"topping\_id": 223,

"time": "2022-04-26T10:51:13.023Z",

"price": 600,

"status": "Delivered"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/order?filter[status]=Delivered

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 124,

"user\_id": 778,

"cart\_id": 112,

"product\_id": 1000,

"payment\_id": 457,

"topping\_id": 223,

"time": "2022-04-26T10:51:13.023Z",

"price": 600,

"status": "Delivered"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)order/123

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 123,

"user\_id": 777,

"cart\_id": 111,

"product\_id": 999,

"payment\_id": 456,

"topping\_id": 222,

"time": "2022-01-26T10:51:13.023Z",

"price": 450,

"status": "New"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

##### Метод POST ресурса /order:

* + - С помощью метода POST можно создать новую сущностью в ресурсе.
    - POST Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| user\_id | integer | - | нет | Идентификатор пользователя |
| cart\_id | integer | - | нет | Идентификатор корзины |
| product\_id | integer | - | нет | Идентификатор продукта |
| payment\_id | integer | - | нет | Идентификатор оплаты |
| topping\_id | integer | - | нет | Идентификатор топпинга |
| time | string | - | нет | Время заказа |
| price | integer | - | нет | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | нет | Статус заказа |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор заказа |
| user\_id | integer | - | да | Идентификатор пользователя |
| cart\_id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| payment\_id | integer | - | да | Идентификатор оплаты |
| topping\_id | integer | - | да | Идентификатор топпинга |
| time | string | - | да | Время заказа |
| price | integer | - | да | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | да | Статус заказа |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример POST параметра

POST [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)order

{

"user\_id": 456789,

"cart\_id": 789456,

"product\_id": 123456,

"payment\_id": 456123,

"topping\_id": 321645,

"time": "2022-01-26T10:51:13.023Z",

"price": 500,

"status": "New"

}

Ответ сервера:

HTTP 201 successful operation

Content-Type: application/json

{

"id": 456978,

"user\_id": 456789,

"cart\_id": 789456,

"product\_id": 123456,

"payment\_id": 456123,

"topping\_id": 321645,

"time": "2022-01-26T10:51:13.023Z",

"price": 500,

"status": "New"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

##### Метод PUT ресурса /order/{orderId}:

* + - С помощью метода PUT можно изменить ресурс.
    - PUT Параметры /{orderId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| orderId | integer | - | да | Идентификатор заказа |

* + - PUT Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| user\_id | integer | - | нет | Идентификатор пользователя |
| cart\_id | integer | - | нет | Идентификатор корзины |
| product\_id | integer | - | нет | Идентификатор продукта |
| payment\_id | integer | - | нет | Идентификатор оплаты |
| topping\_id | integer | - | нет | Идентификатор топпинга |
| time | string | - | нет | Время заказа |
| price | integer | - | нет | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | нет | Статус заказа |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор заказа |
| user\_id | integer | - | да | Идентификатор пользователя |
| cart\_id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| payment\_id | integer | - | да | Идентификатор оплаты |
| topping\_id | integer | - | да | Идентификатор топпинга |
| time | string | - | да | Время заказа |
| price | integer | - | да | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | да | Статус заказа |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PUT параметра

PUT [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)order/123

{

"user\_id": 1,

"cart\_id": 2,

"product\_id": 3,

"payment\_id": 4,

"topping\_id": 5,

"time": "2022-01-26T10:51:13.023Z",

"price": 1000,

"status": "New"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 123,

"user\_id": 1,

"cart\_id": 2,

"product\_id": 3,

"payment\_id": 4,

"topping\_id": 5,

"time": "2022-01-26T10:51:13.023Z",

"price": 1000,

"status": "New"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

##### Метод PATCH ресурса /order/{orderId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{orderId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| orderId | integer | - | да | Идентификатор заказа |

* + - PATCH Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| user\_id | integer | - | нет | Идентификатор пользователя |
| cart\_id | integer | - | нет | Идентификатор корзины |
| product\_id | integer | - | нет | Идентификатор продукта |
| payment\_id | integer | - | нет | Идентификатор оплаты |
| topping\_id | integer | - | нет | Идентификатор топпинга |
| time | string | - | нет | Время заказа |
| price | integer | - | нет | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | нет | Статус заказа |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор заказа |
| user\_id | integer | - | да | Идентификатор пользователя |
| cart\_id | integer | - | да | Идентификатор корзины |
| product\_id | integer | - | да | Идентификатор продукта |
| payment\_id | integer | - | да | Идентификатор оплаты |
| topping\_id | integer | - | да | Идентификатор топпинга |
| time | string | - | да | Время заказа |
| price | integer | - | да | Итоговая сумма заказа |
| status | string | Новый  На паузе  Доставлен  Завершенный | да | Статус заказа |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)order/123

{

"user\_id": 777,

"cart\_id": 111,

"product\_id": 999,

"payment\_id": 456,

"topping\_id": 222,

"time": "2022-01-26T10:51:13.023Z",

"price": 450,

"status": "New"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 123,

"user\_id": 777,

"cart\_id": 111,

"product\_id": 999,

"payment\_id": 456,

"topping\_id": 222,

"time": "2022-01-26T10:51:13.023Z",

"price": 450,

"status": "New"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

##### Метод DELETE ресурса /order/{orderId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{orderId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| orderId | integer | - | да | Идентификатор заказа |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/order/123

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 order not found

Content-Type: application/json

{

"code": 404,

"message": "order not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /stock:

##### Метод GET ресурса /stock:

* + - С помощью метода GET можно получить данные по корзинам, отфильтровать и отсортировать результаты.
    - GET Параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |
| date\_from | string | - | нет | Начальная дата выборки |
| date\_to | string | - | нет | Конечная дата выборки |
| category | string | product topping | нет | Категория, по которой осуществлялась выборка |

* + - Возвращаемые параметры ресурса stock:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| name | string | - | да | Название остатка |
| category | string | продукт  топпинг | да | Категория остатка. Может быть продуктом или топпингом |
| number\_of\_stocks | integer | - | да | Кол-во остатков |
| date | string | - | да | Дата, за которую осуществлялась выборка |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)stock

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC cheese",

"category": "topping",

"number\_of\_stocks": 50,

"date": "2022-01-01T01:18:50.385Z"

},

{

"name": "STC bacon",

"category": "product",

"number\_of\_stocks": 10,

"date": "2022-12-31T11:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/stock?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC bacon",

"category": "product",

"number\_of\_stocks": 10,

"date": "2022-12-31T11:18:50.385Z"

},

{

"name": "STC cheese",

"category": "topping",

"number\_of\_stocks": 50,

"date": "2022-01-01T01:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

Пример GET параметра с фильтрацией

GET https://api.pizzeria.com/data/2.5/v1/stock?filter[category]=topping

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC cheese",

"category": "topping",

"number\_of\_stocks": 50,

"date": "2022-01-01T01:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

Пример GET параметра date\_from

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)stock?date\_from=2022-01-01

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC bacon",

"category": "product",

"number\_of\_stocks": 10,

"date": "2022-12-31T11:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

Пример GET параметра date\_to

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)stock?date\_to=2022-10-01

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC cheese",

"category": "topping",

"number\_of\_stocks": 50,

"date": "2022-01-01T01:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

Пример GET параметра date\_to и date\_from

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)stock?date\_from=2022-01-01&date\_to=2022-10-01

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC cheese",

"category": "topping",

"number\_of\_stocks": 50,

"date": "2022-01-01T01:18:50.385Z"

},

{

"name": "STC bacon",

"category": "product",

"number\_of\_stocks": 10,

"date": "2022-12-31T11:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

Пример GET параметра category

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)stock?category=product

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"name": "STC bacon",

"category": "product",

"number\_of\_stocks": 10,

"date": "2022-12-31T11:18:50.385Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 stock not found

Content-Type: application/json

{

"code": 404,

"message": "stock not found"

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Ресурс /payment:

##### Метод GET ресурса /payment:

* + - С помощью метода GET можно получить данные по корзинам, отфильтровать и отсортировать результаты.
    - GET Параметры :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| sort | string | asc desc | нет | Сортировка результатов в порядка возрастания или убывания |
| filter | string | - | нет | Фильтрация результатов |

* + - GET Параметры /{paymentId} :

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| paymentId | integer | - | да | Идентификатор оплаты |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор оплаты заказа |
| amount | integer | - | да | Сумма оплаты заказа |
| method | string | Карта  Наличные  Перевод онлайн  Картой по терминалу | да | Метод оплаты заказа |
| time | string | - | да | Время оплаты заказа |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример GET метода

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)payment

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 456,

"amount": 500,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

},

{

"id": 457,

"amount": 600,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 payment not found

Content-Type: application/json

{

"code": 404,

"message": "payment not found"

}

Пример GET параметра с сортировкой

GET [https://api.pizzeria.com/data/2.5/v1/payment?sort=](https://api.mamamiapizzeria.com/data/2.5/v1/user?sort=asc)asc

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 456,

"amount": 500,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

},

{

"id": 457,

"amount": 600,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 payment not found

Content-Type: application/json

{

"code": 404,

"message": "payment not found"

}

Пример GET параметра с фильтрацией

GET [https://api.pizzeria.com/data/2.5/v1/order?filter[method]=Credit](https://api.pizzeria.com/data/2.5/v1/order?filter=method=Credit)%20Card

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

[

{

"id": 456,

"amount": 500,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

},

{

"id": 457,

"amount": 600,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

}

]

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 payment not found

Content-Type: application/json

{

"code": 404,

"message": "payment not found"

}

Пример GET параметра с ID

GET [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user?filter=new)payment/456

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 456,

"amount": 500,

"method": "Credit Card",

"time": "2022-04-26T13:13:02.868Z"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 payment not found

Content-Type: application/json

{

"code": 404,

"message": "payment not found"

}

##### Метод PATCH ресурса /payment/{paymentId}:

* + - С помощью метода PATCH можно частично изменить ресурс.
    - PATCH Параметры /{paymentId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| paymentId | integer | - | да | Идентификатор оплаты |

* + - PATCH Параметры:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| amount | integer | - | нет | Сумма оплаты заказа |
| method | string | Карта  Наличные  Перевод онлайн  Картой по терминалу | нет | Метод оплаты заказа |
| time | string | - | нет | Время оплаты заказа |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| id | integer | - | да | Идентификатор оплаты заказа |
| amount | integer | - | да | Сумма оплаты заказа |
| method | string | Карта  Наличные  Перевод онлайн  Картой по терминалу | да | Метод оплаты заказа |
| time | string | - | да | Время оплаты заказа |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример PATCH параметра

PATCH [https://api.pizzeria.com/data/2.5/v1/](https://api.mamamiapizzeria.com/data/2.5/v1/user)payment/456

{

"amount": 600,

"method": "Cash",

"time": "2022-04-26T13:13:02.868Z"

}

Ответ сервера:

HTTP 200 successful operation

Content-Type: application/json

{

"id": 456,

"amount": 600,

"method": "Cash",

"time": "2022-04-26T13:13:02.868Z"

}

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 payment not found

Content-Type: application/json

{

"code": 404,

"message": "payment not found"

}

##### Метод DELETE ресурса /payment/{paymentId}:

* + - С помощью метода DELETE можно удалить ресурс.
    - DELETE Параметры /{paymentId}:

| Название параметра | Тип поля | Доступные значения | Обязательно присутствует в запросе | Описание |
| --- | --- | --- | --- | --- |
| paymentId | integer | - | да | Идентификатор оплаты |

* + - Возвращаемые параметры:

| Название параметра | Тип параметра | Доступные значения | Обязательно присутствует в ответе | Описание |
| --- | --- | --- | --- | --- |
| - | - | - | - | - |

**Пример работы с ресурсом**

Суффиксы:

* .json - получить ответ с сервера в формате JSON
* .xml - получить ответ с сервера в формате XML

Пример DELETE параметра

DELETE https://api.pizzeria.com/data/2.5/v1/payment/456

Ответ сервера:

HTTP 200 successful operation

Отрицательный ответ сервера:

HTTP 400 invalid input

Content-Type: application/json

{

"code": 400,

"message": "invalid input"

}

ИЛИ

HTTP 404 payment not found

Content-Type: application/json

{

"code": 404,

"message": "payment not found"

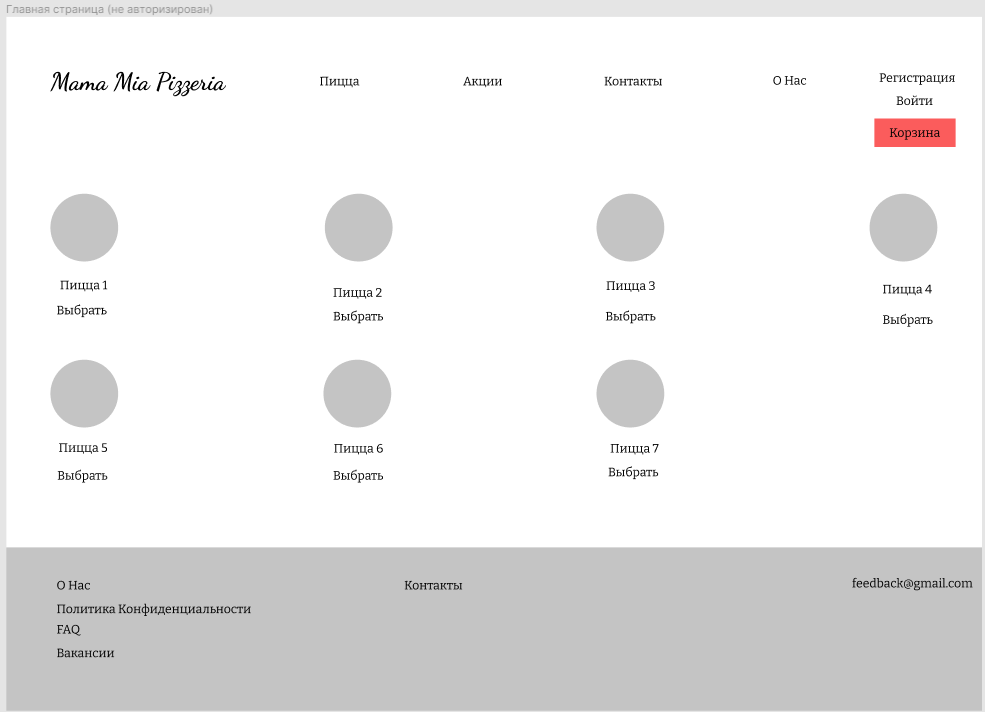
}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

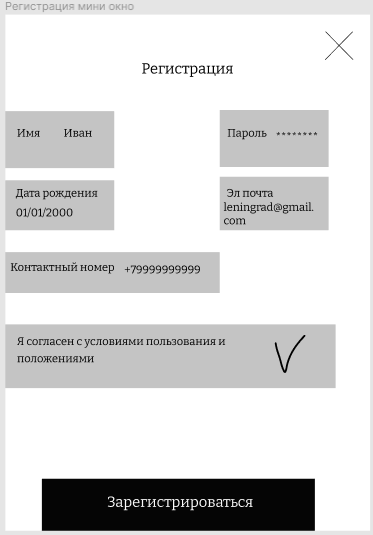
### **Пользовательские интерфейсы**

*[Структура и описание экранов пользовательского интерфейса.]*

*[Очень полезно добавлять макеты и снабжать их кратким описанием экранов.]*



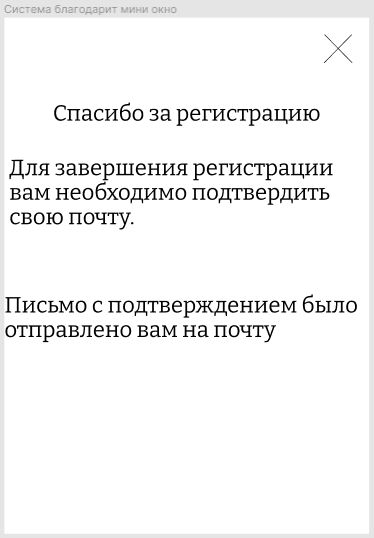
Главная страница неавторизованного пользователя. На экране есть меню пицц, ссылки на акции, контакты, о нас. Также есть ссылки на регистрацию, авторизацию и корзину.



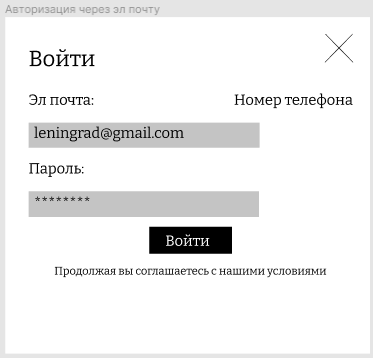
При регистрации, пользователь видит поля для ввода данных и условия пользования и положения, напротив которой, пользователь должен поставить галочку, давая свое согласие.



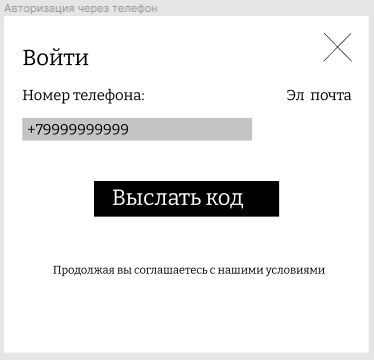
Альтернативный вид регистрации.



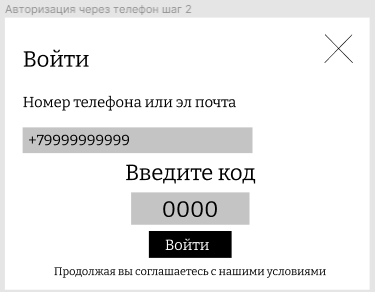
Окончание регистрации. Пользователь видит такое сообщение от системы после прохождения регистрации.



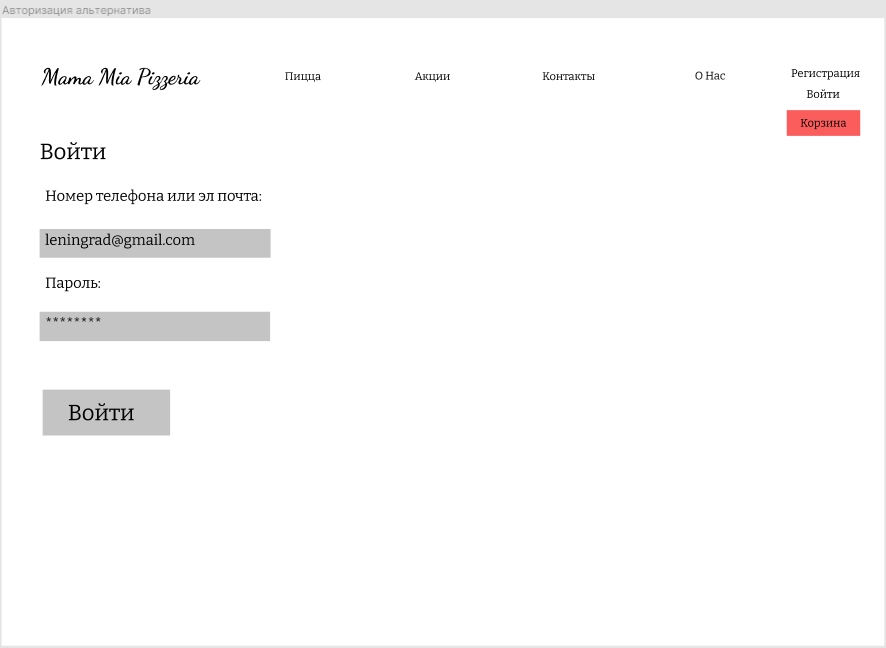
Авторизация может проходить по электронной почте. Есть поля для электронной почты и пароля. В данном случае, пользователь соглашается с условиями сайта автоматически путем авторизации или входа на сайт.



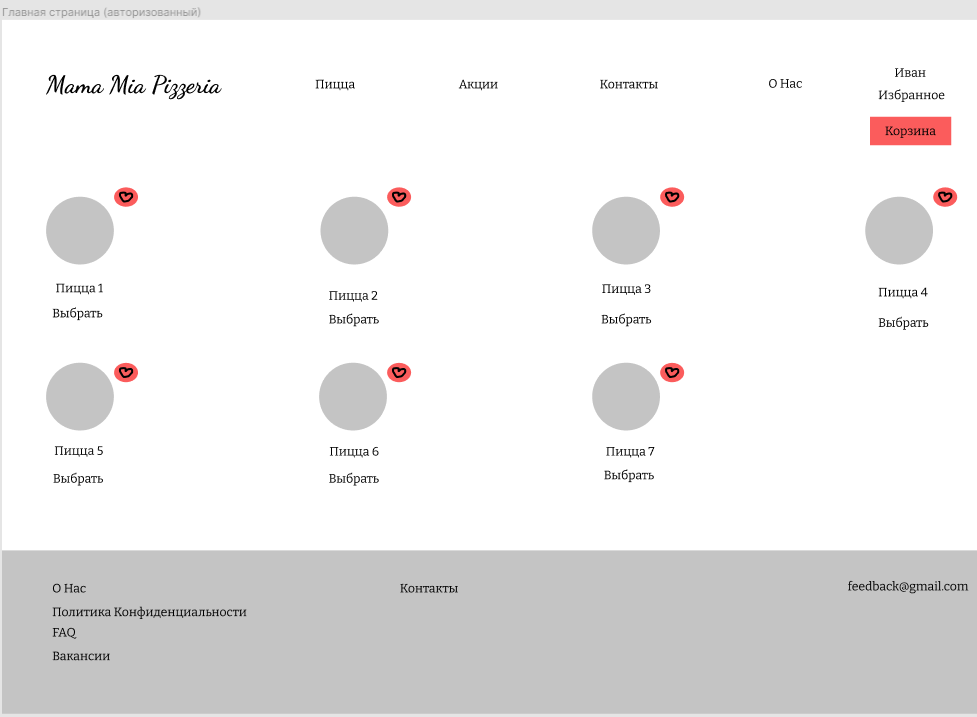
Также авторизация может проходить по номеру телефона. Пользователь вводит свой зарегистрированный номер телефона и переходит на следующий этап авторизации.



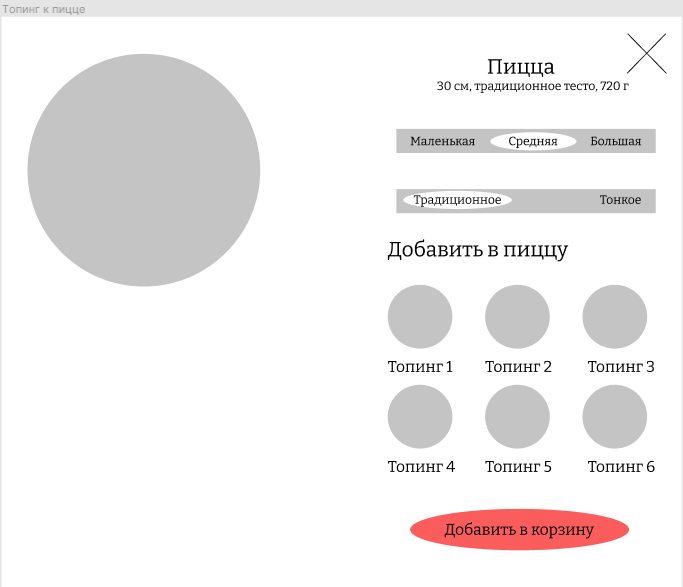
Тут пользователь получает код подтверждения на свой телефон и вписывает этот код в поле. После, пользователь соглашается с условиями сайта автоматически путем авторизации или входа на сайт.



Альтернатива авторизации в мини окнах. Если ссылки на основные страницы как на главной странице.



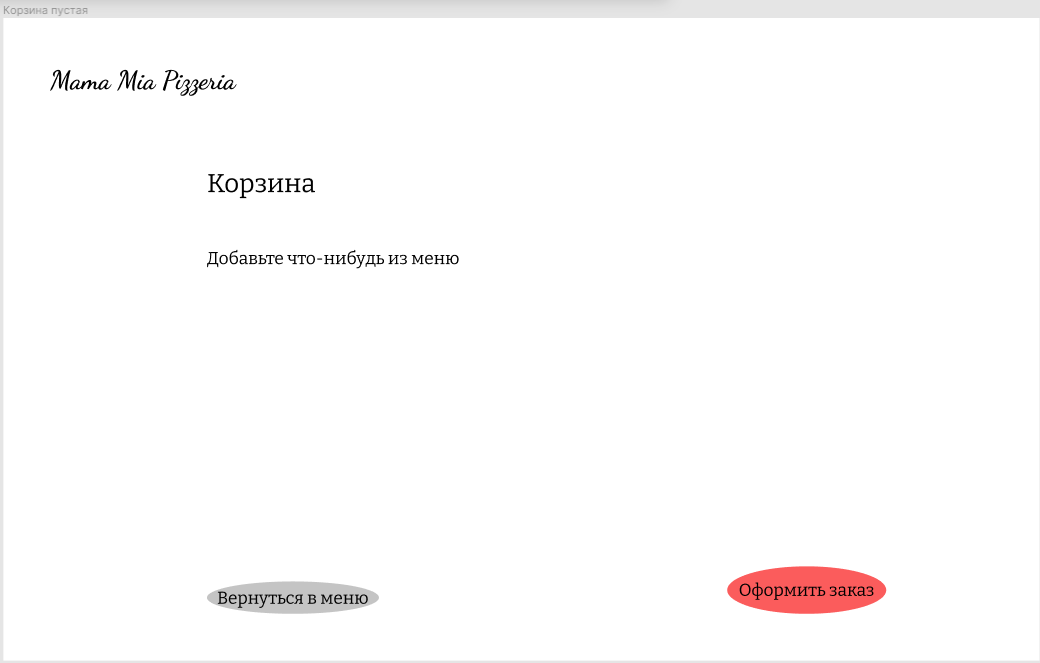
Это уже главная страница авторизированного пользователя. Практически тоже самое, за исключением опции добавить продукты в избранное и самой ссылки на избранные продукты. Вместо регистрации, авторизованный пользователь видит свое имя. Если ссылки на основные страницы как на главной странице.



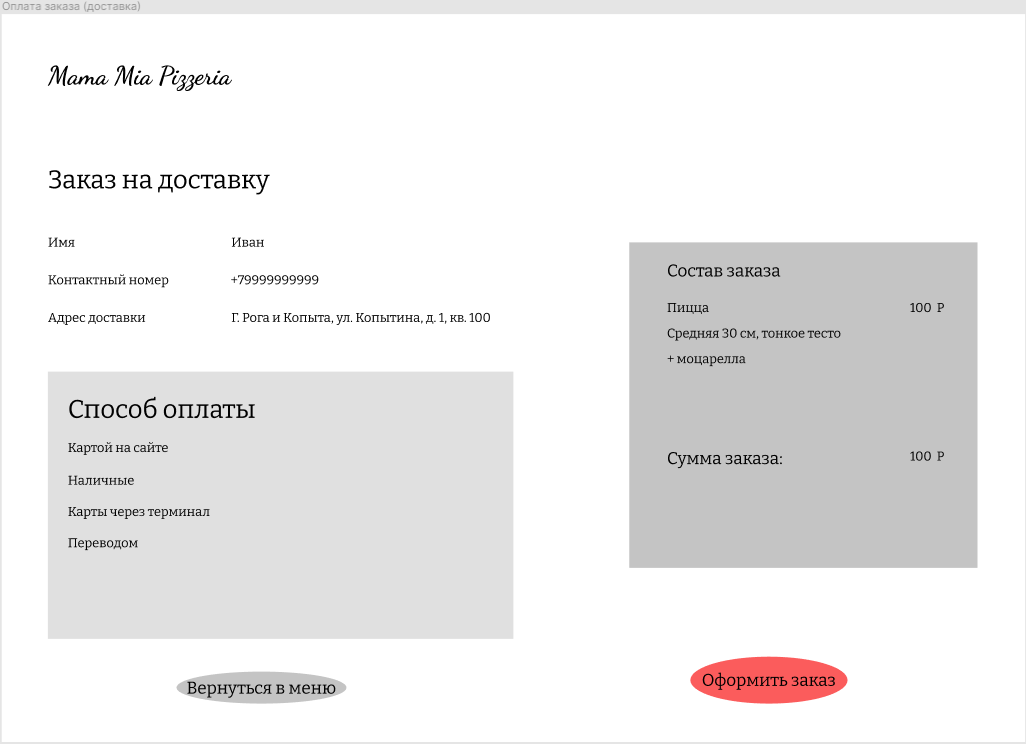
После выбора пиццы, пользователь видит параметры выбранного продукта. Есть опции размера и толщины теста, также можно выбрать топпинги для пиццы. Возможность добавить в корзину продукт можно сразу, выделение красным способствует сфокусированности пользователя на продолжении процесса оформления заказа.



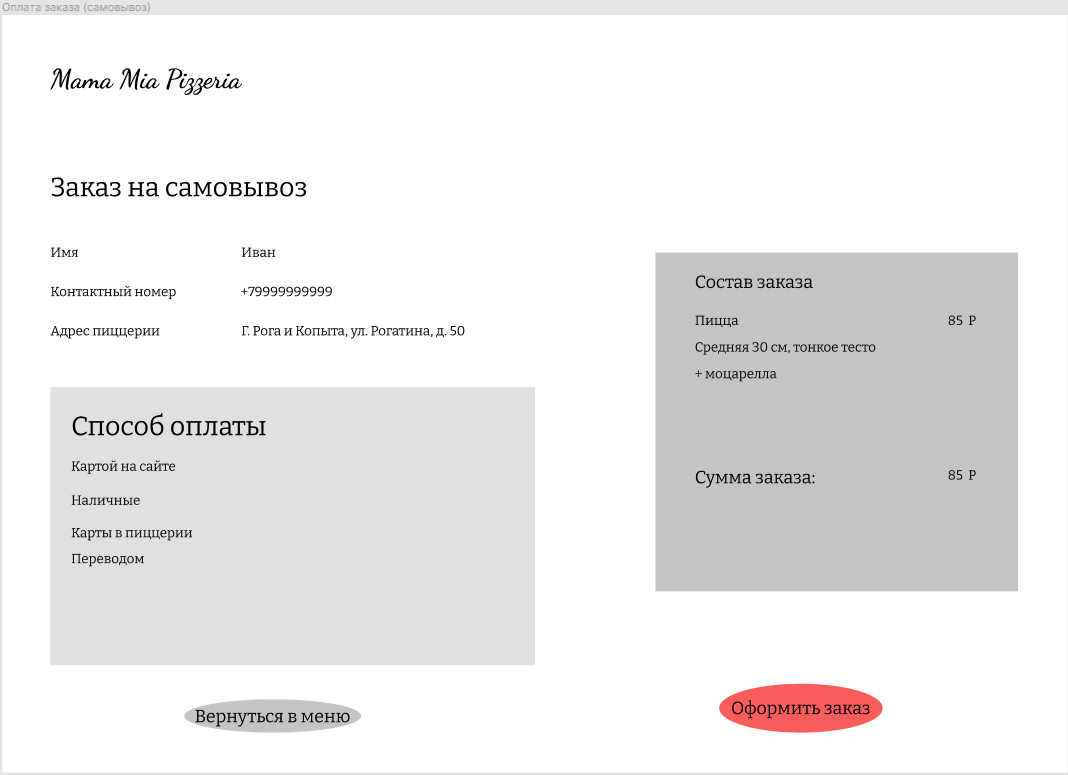
После добавления продукта в корзину, пользователь переходит в корзину для подтверждения своего выбора. На экране присутствует сам продукт с выбранными опциями, его количеством и стоимостью. Есть возможности удаления пиццы из корзины, вернуться в главное меню и оформить заказ. Красный цвет поможет пользователю продолжить процесс оформления заказа.



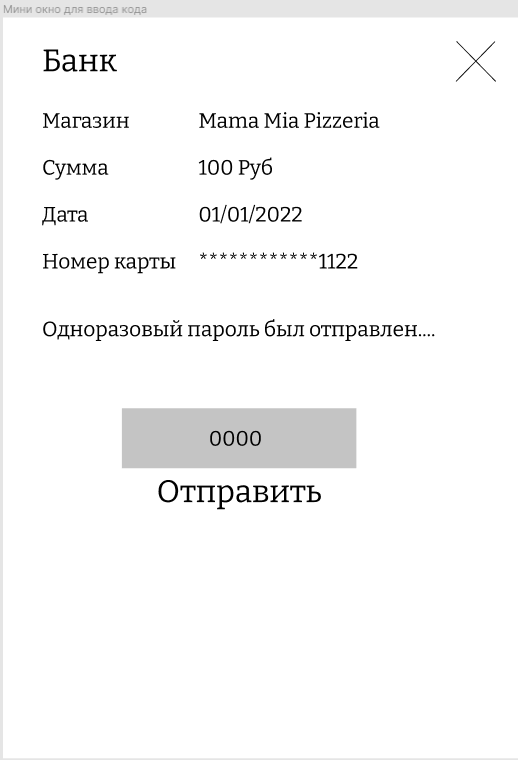
Это вид пустой корзины.



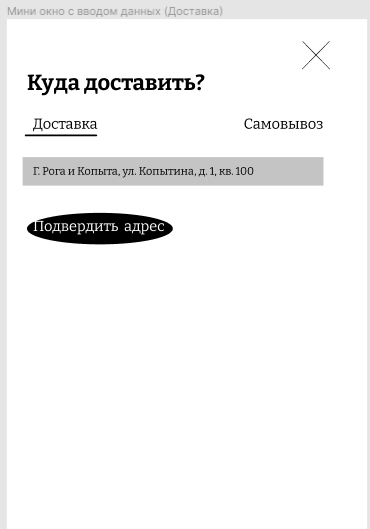
Этот экран содержит всю информацию о заказе, пользователе и способе оплаты при выборе доставки заказа на дом.



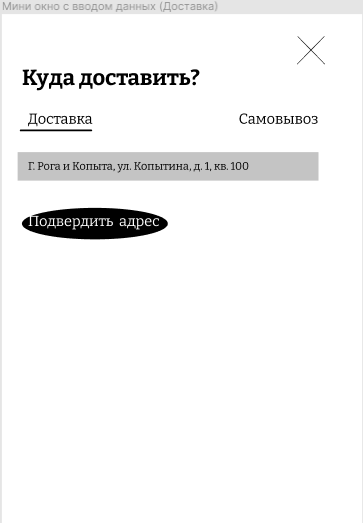
Так выглядит экран для самовывоза.



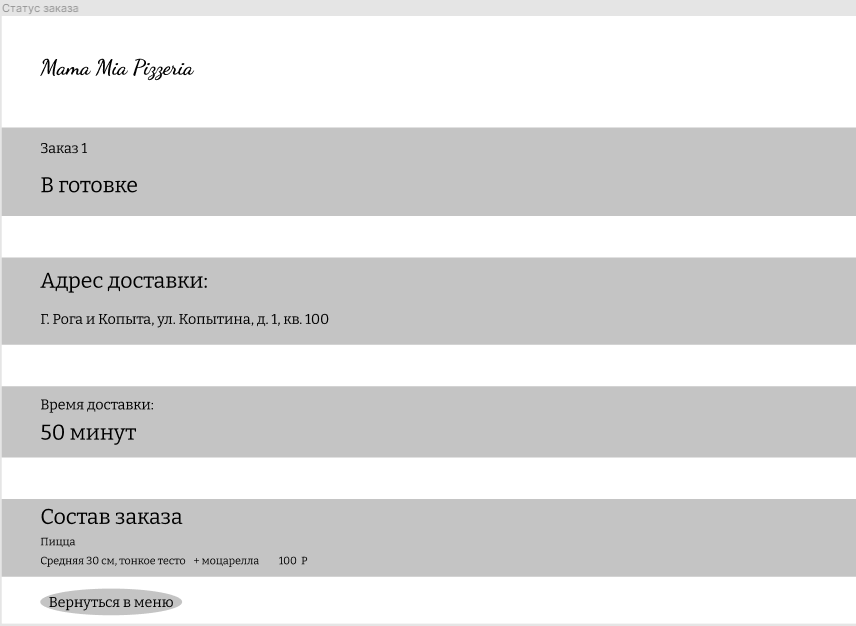
При доставке заказа на дом, по умолчанию проходит оплата картой. Такое мини окно появляется при оплате через банк и со стороны банка.



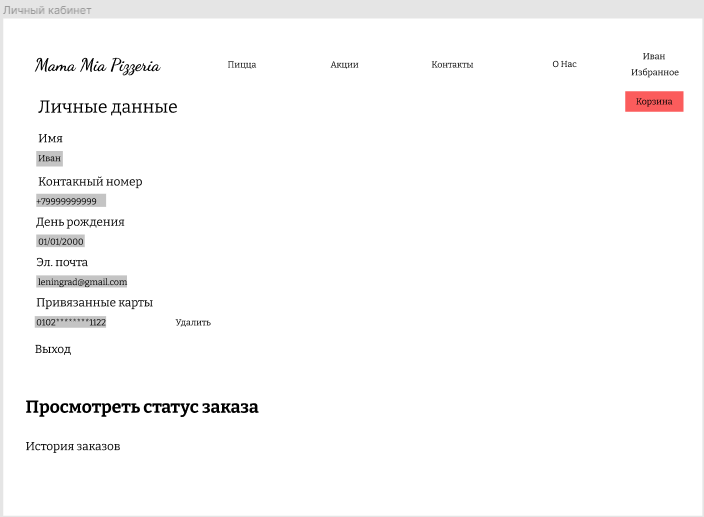
Этот экран содержит адрес доставки заказа. Здесь же пользователь подтверждает свой адрес и при желании может его изменить. Также здесь присутствует возможность самовывоза.



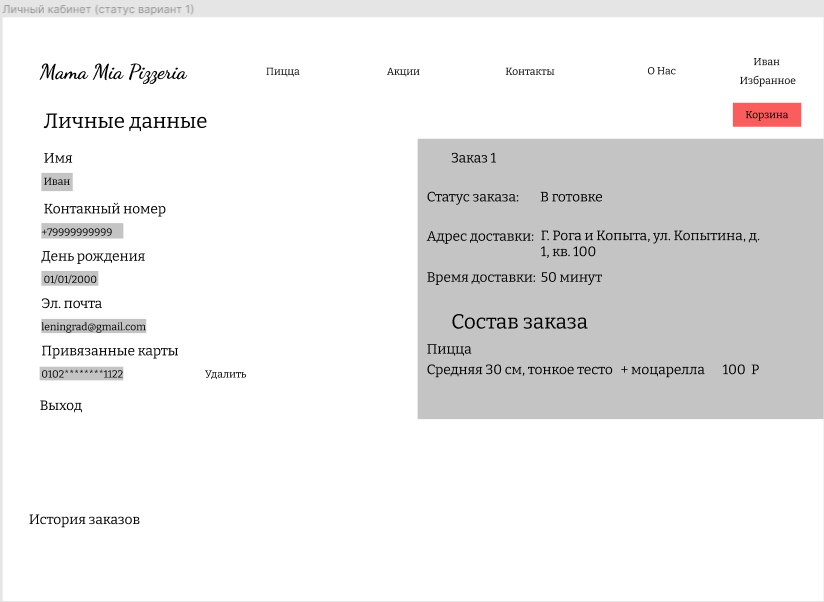
Это экран самовывоза с улицей заведения.



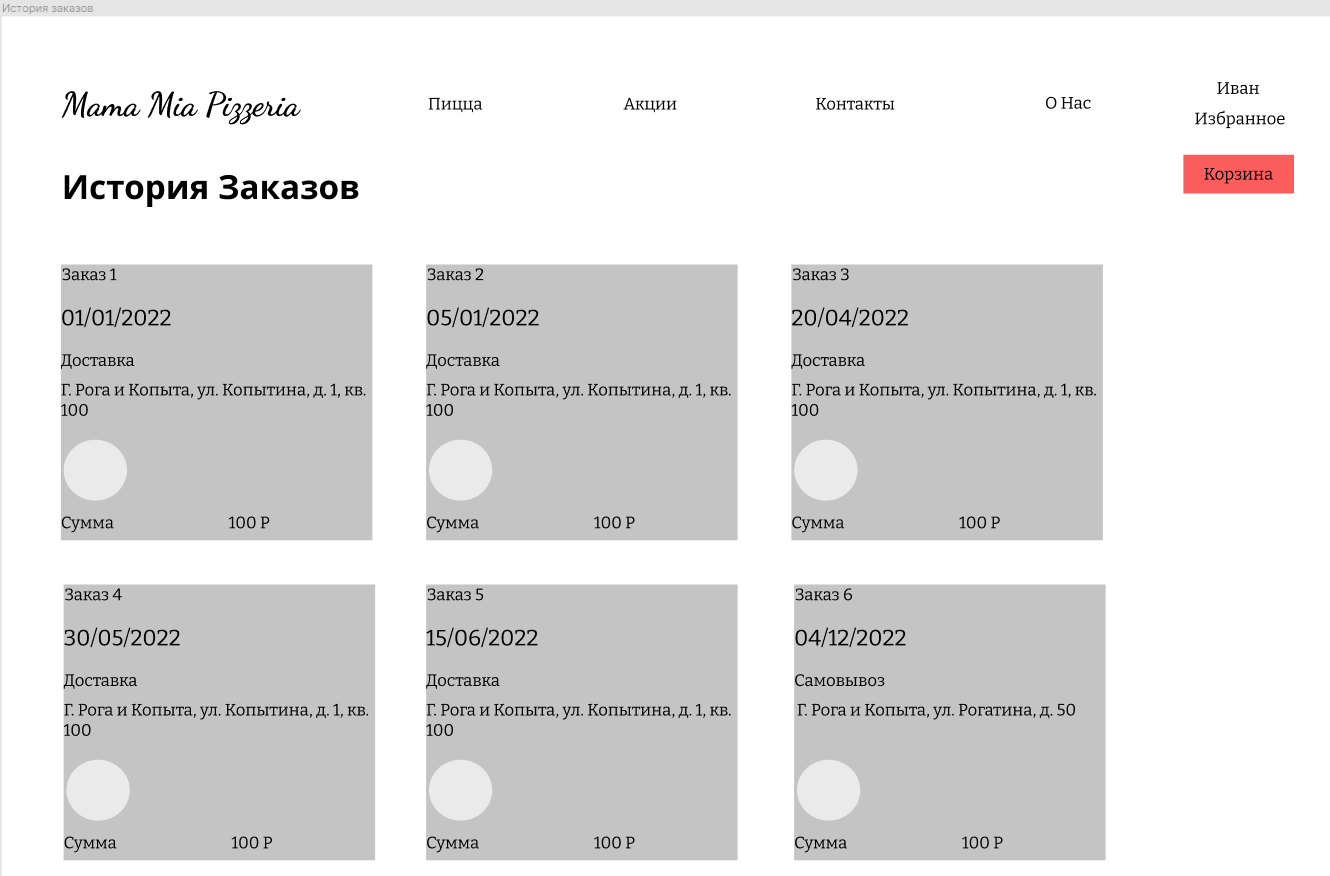
Экран статуса заказа, адреса доставки, приблизительного времени доставки и состава заказа.



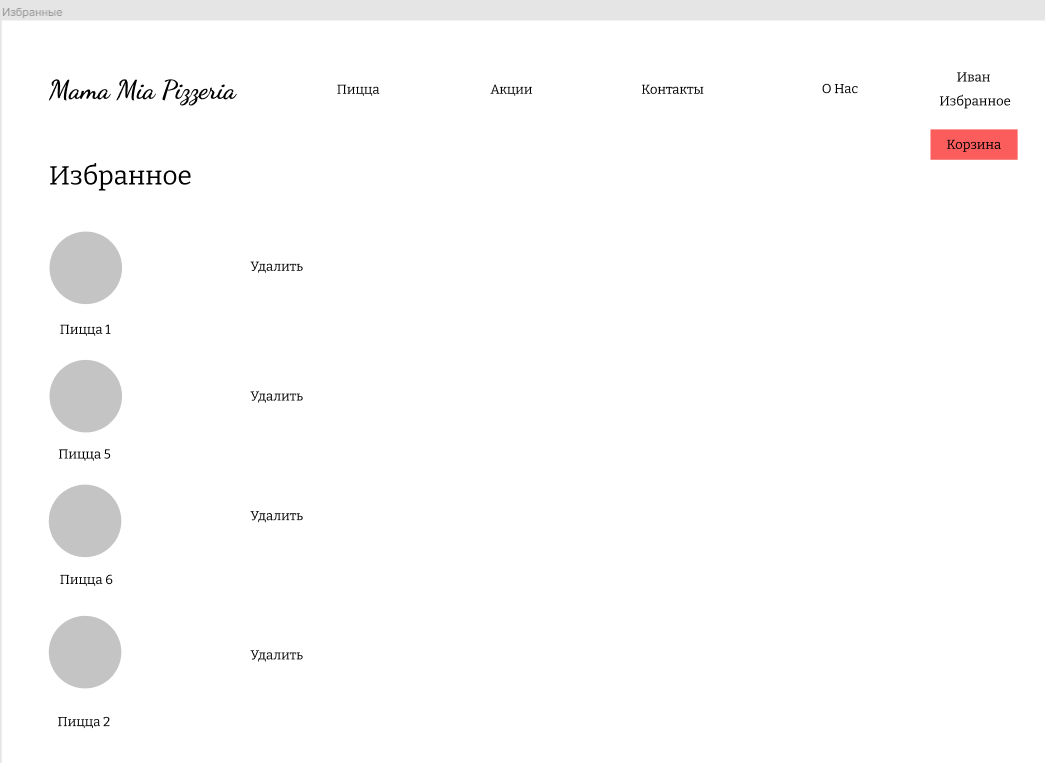
Экран личного кабинета пользователя. Там имеется вся информация о пользователе, которую можно редактировать при желании. Есть возможность просмотреть историю заказов. Если ссылки на основные страницы как на главной странице.



Альтернативный вид личного кабинета. Данные пользователя + статус заказа на экране.



История заказов пользователя. Присутствует вся информация по прошлым заказам, улица доставки и самовывоза, дата заказа, номер заказа, сумма, тип заказа - доставка или самовывоз. Если ссылки на основные страницы как на главной странице.



Этот экран отображает список избранных продуктов с возможностью удаления последнего. Если ссылки на основные страницы как на главной странице.

Ссылка на прототип:

[https://www.figma.com/file/Eb3bRRXwxwwxHgtAVw5ltj/Магазин-Пиццы?node-id=0%3A1](https://www.figma.com/file/Eb3bRRXwxwwxHgtAVw5ltj/%D0%9C%D0%B0%D0%B3%D0%B0%D0%B7%D0%B8%D0%BD-%D0%9F%D0%B8%D1%86%D1%86%D1%8B?node-id=0%3A1)

Карта сайта:

https://docs.google.com/document/d/1hXHe6J\_Up6cwuX23SAb-r5UNJt-pj\_BlKK48BKXOMGo/edit

### **Удобство использования**

*[Должны быть конкретными, если их невозможно сформулировать — раздел не пишут.]*

1. Цвета и шрифты пользовательского интерфейса веб-сайта должны соответствовать требованиям стандарта по инклюзивному дизайну WCAG 2.0
2. Веб сайт должен быть построен на фундаментальных технологиях XML и CSS
3. Разработка веб-сайта должна подходить стандартам JavaScript Web API используя DOM & XBL

### **Атрибуты качества**

[Остальные нефункциональные требования, описывающие качества работы системы: производительность, надежность, доступность, информационная безопасность, поддерживаемость и другие. Нужно стараться указывать измеримые характеристики.]

1. Система должна обрабатывать 95% запросов от пользователей не дольше 1 секунду.
2. Система должна обеспечивать доступность на уровне 99,9% времени работы.
3. Система должна обеспечивать количество одновременно работающих пользователей от 10 человек
4. Система должна поддерживать частоту запросов от пользователя в минуту меньше или равно 4
5. Система должна обрабатывать количество исполняемых транзакций в секунду от 3
6. Система должна обеспечить коэффициент количества операций / (секунду \* стоимость оборудования, k$) не менее 3
7. Система должна иметь вероятность сбоя (без ущерба безопасности) 5%
8. Система должна иметь время восстановления после сбоя меньше или равной 5 минутам
9. Система должна иметь продолжительность простоя системы 1 минуты в час
10. Система должна иметь вероятность утечки данных в результате сбоя менее 0.1%
11. Система должна соответствовать стоимости десятикратного увеличения мощности системы между 400-900%
12. Система должна загружать страницу не больше 2-3 секунд.