

Интернет-магазин

Дизайн Решения

swagger: "2.0"

info:

description: "This is a sample server Book store server"

version: "1.0.0"

title: "Sychev Fedor Book Store"

host: "userstore.swagger.io"

basePath: "/v2"

tags:

- name: "authorization"

description: "Everything about users accounts"

- name: "cart"

description: "Access to user cart"

- name: "catalog"

description: "Operations about user"

schemes:

- "https"

- "http"

paths:

/authorization:

post:

tags:

- "authorization"

summary: "Add a new user"

description: ""

operationId: "addUser"

consumes:

- "application/json"

- "application/xml"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "body"

name: "body"

description: "User object that needs to be added to the store"

required: true

schema:

$ref: "#/definitions/User"

responses:

"405":

description: "Invalid input"

/authorization/{userID}:

get:

tags:

- "authorization"

summary: "Find user by ID"

description: "Returns a single pet"

operationId: "getPetById"

produces:

- "application/xml"

- "application/json"

parameters:

- name: "userID"

in: "path"

description: "ID of pet to return"

required: true

type: "integer"

format: "int64"

responses:

"200":

description: "successful operation"

schema:

$ref: "#/definitions/User"

"400":

description: "Invalid ID supplied"

"404":

description: "User not found"

post:

tags:

- "authorization"

summary: "Updates a user with form data"

description: ""

operationId: "updateUserWithForm"

consumes:

- "application/x-www-form-urlencoded"

produces:

- "application/xml"

- "application/json"

parameters:

- name: "userID"

in: "path"

description: "ID of user that needs to be updated"

required: true

type: "integer"

format: "int64"

- name: "name"

in: "formData"

description: "Updated name of the user"

required: false

type: "string"

- name: "password"

in: "formData"

description: "Updated password of the user"

required: false

type: "string"

responses:

"405":

description: "Invalid input"

/cart/{userID}:

get:

tags:

- "cart"

summary: "Find cart products by userID"

description: "Returns a single cart"

operationId: "getCartById"

produces:

- "application/xml"

- "application/json"

parameters:

- name: "userID"

in: "path"

description: "ID of user to return"

required: true

type: "integer"

format: "int64"

responses:

"200":

description: "successful operation"

schema:

$ref: "#/definitions/Cart"

"400":

description: "Invalid ID supplied"

"404":

description: "Cart not found"

/cart/user/{cartId}:

post:

tags:

- "cart"

summary: "Find cart order by ID"

description: "For valid response try integer IDs with value >= 1 and <= 10. Other values will generated exceptions"

operationId: "getCartByIdForUpdate"

produces:

- "application/xml"

- "application/json"

parameters:

- name: "cartId"

in: "path"

description: "ID of cart that needs to be fetched"

required: true

type: "integer"

maximum: 10.0

minimum: 1.0

format: "int64"

responses:

"200":

description: "successful operation"

schema:

$ref: "#/definitions/Cart"

"400":

description: "Invalid ID supplied"

"404":

description: "Cart not found"

/catalog:

post:

tags:

- "catalog"

summary: "Create catalog"

description: "Creates catalog"

operationId: "createCatalog"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "body"

name: "body"

description: "Created user object"

required: true

schema:

$ref: "#/definitions/Catalog"

responses:

default:

description: "successful operation"

/catalog/createWithList:

post:

tags:

- "catalog"

summary: "Creates list of products with given input array"

description: ""

operationId: "createCatalogWithProductsList"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "body"

name: "body"

description: "List of Product object"

required: true

schema:

type: "array"

items:

$ref: "#/definitions/Catalog"

responses:

default:

description: "successful operation"

/catalog/{catalogname}:

get:

tags:

- "catalog"

summary: "Get catalog by catalog name"

description: ""

operationId: "getCatalogByName"

produces:

- "application/xml"

- "application/json"

parameters:

- name: "catalogname"

in: "path"

description: "The name that needs to be fetched. Use catalog1 for testing. "

required: true

type: "string"

responses:

"200":

description: "successful operation"

schema:

$ref: "#/definitions/Catalog"

"400":

description: "Invalid catalogname supplied"

"404":

description: "Catalog not found"

delete:

tags:

- "catalog"

summary: "Delete catalog"

description: "This can only be done by the logged in user."

operationId: "deleteCatalog"

produces:

- "application/xml"

- "application/json"

parameters:

- name: "catalogname"

in: "path"

description: "The name that needs to be deleted"

required: true

type: "string"

responses:

"400":

description: "Invalid catalogname supplied"

"404":

description: "Catalog not found"

securityDefinitions:

petstore\_auth:

type: "oauth2"

authorizationUrl: "http://petstore.swagger.io/oauth/dialog"

flow: "implicit"

scopes:

write:pets: "modify pets in your account"

read:pets: "read your pets"

api\_key:

type: "apiKey"

name: "api\_key"

in: "header"

definitions:

Cart:

type: "object"

properties:

id:

type: "integer"

format: "int64"

userId:

type: "integer"

format: "int64"

quantity:

type: "integer"

format: "int32"

created\_at:

type: "string"

format: "date-time"

status:

type: "string"

description: "Order Status"

enum:

- "placed"

- "approved"

- "delivered"

complete:

type: "boolean"

default: false

xml:

name: "Order"

Catalog:

type: "object"

properties:

id:

type: "integer"

format: "int64"

name:

type: "string"

xml:

name: "Catalog"

User:

type: "object"

properties:

id:

type: "integer"

format: "int64"

username:

type: "string"

email:

type: "string"

password:

type: "string"

address:

type: "string"

xml:

name: "User"

externalDocs:

description: "Find out more about Swagger"

url: "http://swagger.io"

Спецификации:

1) POST /authorization Добавляет пользователя в базу данных по логину, паролю, электронной почте и адресу

2) GET /authorization/{userID} Авторизует пользователя в системе по логину и паролю

3) POST /authorization/{userID} Изменяет данные пользователя на заданные новые

4) POST /cart Создает корзину пользователю

5) GET /cart/{userID} Осуществляет поиск корзины по пользователю

6) POST /catalog Создает каталог товаров с определенным именем

7) POST /catalog/createWithList создает каталог с определенным именем и вносит туда товары из списка

8) GET catalog/{catalogname} Осуществляет поиск каталога по имени

9) DELETE catalog/{catalogname} Осуществляет поиск каталога по имени и удаляет его

**Структура данных API сервиса авторизации:**

**Заголовок запроса**

В заголовке запроса обязательно указывается:

1. адрес сервера:
   1. для команд аутентификации — <Адрес для аутентификации>
   2. для прочих команд — <Адрес для остальных команд>;
2. идентификатор сессии «SessionID»;
3. кодировка символов в поле «Content-Type»:
   1. рекомендуем «utf-8» — application/json-rpc;charset=utf-8;
   2. либо «windows-1251» — application/json-rpc;charset=windows-1251;
4. название и версия информационной системы в заголовке «User-Helper».

**Объект запроса**

Объект запроса содержит:

* значение и версию спецификации JSON-RPC — «jsonrpc»:«2.0»;
* параметр «method» с названием команды;
* параметр «params» с объектом параметров команды;
* параметр «id»с целочисленным значением.

**Объект ответа**

В ответ на запрос с сервера возвращается JSON-структура:

* значение и версия спецификации JSON-RPC — «jsonrpc»:«2.0»;
* параметр «id» со значением из объекта запроса;
* значение объекта «result», содержащее ответ на команду.

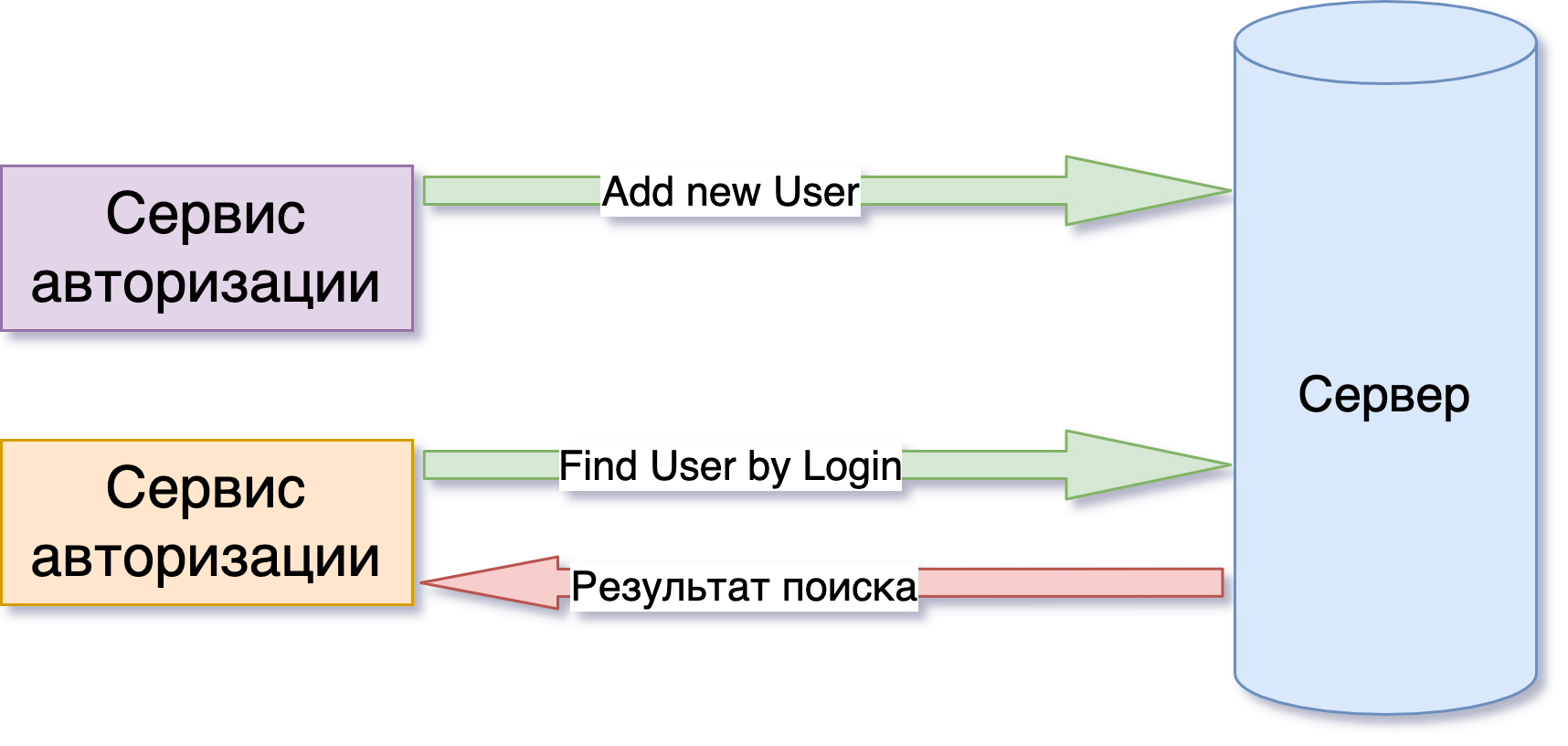
Если произошла ошибка, в ответе на запрос вернется «error».

**Объект ошибки**

Если вернулась ошибка, в объекте ответа в поле «error» будут указаны:

* «code» — код ошибки;
* «message» — текст ошибки;
* «details» — подробное описание;
* «data» — тип ошибки.

Алгоритм работы сервиса:



Подготовлено: Сычев Ф. В.

Проект: Внедрение интернет-магазина в компании Издательский Дом

# Оглавление

[**Оглавление**](#_h2rq0xjczs5k) **2**

[**Бизнес-цель**](#_vfj0t65hkaqh) **4**

[**Бизнес-требования**](#_xai8cpvd5us8) **5**

[**Описание бизнес-процесса**](#_ucwp3wyf3l1m) **6**

[**Архитектура решения**](#_jerixkzfuh13) **7**

[**Сервис каталога продуктов**](#_lk51g0ks0fwq) **8**

[Функциональные требования](#_cmnaysb8jwdr) 8

[Нефункциональные требования](#_hyykwr4b2k95) 8

[Структура данных](#_3y06hkbl6l4k) 8

[Алгоритмы](#_o3i8thgxmo1f) 8

[Каталог](#_zai9eojzapd5) 8

[Поиск по автору](#_ooqukwmu32gq) 8

[Стек используемых технологий](#_a6jxcq7nqtcj) 9

[**Сервис корзины**](#_79ck44v3q8tc) **10**

[Функциональные требования](#_2xzhzoolvq76) 10

[Нефункциональные требования](#_g3gqpajg92ok) 10

[Структура данных](#_w4zun5zds8w9) 10

[Алгоритмы](#_8csxjulprb9e) 10

[Стек используемых технологий](#_lpgfk8samvb1) 10

[**Сервис авторизации**](#_sn7i14i7uqvt) **11**

[Функциональные требования](#_o1sdqyboqn57) 11

[Нефункциональные требования](#_q8by62wh2z59) 11

[Структура данных](#_8c5dp1tr7xwu) 11

[Алгоритмы](#_2epxqdurzcf) 11

[Стек используемых технологий](#_ib7er1rfk0hb) 11

[**Frontend**](#_gw6ncqprb49a) **12**

[Функциональные требования](#_5k6y0xpq8e7g) 12

[Нефункциональные требования](#_b47eprba7flv) 12

[Структура данных](#_1rui2bgqc843) 12

[Алгоритмы](#_rrkelykubz31) 12

[Экранные формы](#_kqqn4obgepx7) 12

[Форма входа](#_yro4izw9ved3) 12

[Каталог](#_joniftqqzi73) 12

[Корзина](#_92c7nu2zi7gh) 12

[Стек используемых технологий](#_wptolvap4vrd) 12

[**Интеграция**](#_3lqul3g5z5ja) **12**

[Номенклатура (1С УПП ➡ Сервис каталога)](#_5qxiubrayz15) 13

[Соответствие полей](#_7p36oq1uipue) 13

[Триггеры запуска](#_r9wjk0qdnzar) 13

[Алгоритмы](#_bolwwgxflvz2) 13

[Спецификация OpenAPI (Swagger)](#_83uqn4u6y6tz) 13

[Клиенты (Сервис корзины ➡ CRM)](#_165s0e2jxcm4) 13

[Соответствие полей](#_1oq3l2f2rxj0) 13

[Триггеры запуска](#_9g7weymrp60t) 13

[Алгоритмы](#_95spjw4i3jzr) 13

[Спецификация OpenAPI (Swagger)](#_admjmenenroz) 13

[Каталог (Сервис каталога ➡ Frontend)](#_u5ko13b8a07t) 13

[Соответствие полей](#_mk5y39wcg2gp) 13

[Триггеры запуска](#_mh0oicw7tozm) 14

[Алгоритмы](#_b4h4lb0zo3r) 14

[Спецификация OpenAPI (Swagger)](#_3fx8rvotmvev) 14

[**Ролевая модель**](#_gky9v2eq8wzl) **15**

[Роли](#_i641zxckhkpi) 15

[Матрица доступа](#_ltuzl4qf3ud4) 15

[**Система логирования**](#_xbzo22jtiwyq) **16**

[Уровни логирования](#_i9yy0mv0dnzl) 16

[Состав логов](#_a1caeiptxkdy) 16

[Схема хранения логов (в днях)](#_j0crfs30ip8g) 16

[**Система мониторинга**](#_tdqcrm1zsc5d) **17**

[**Стратегия резервного копирования**](#_jas7832hxhmd) **18**

# Бизнес-цель

Необязательно. По материалам уроков БА.

.

# Бизнес-требования

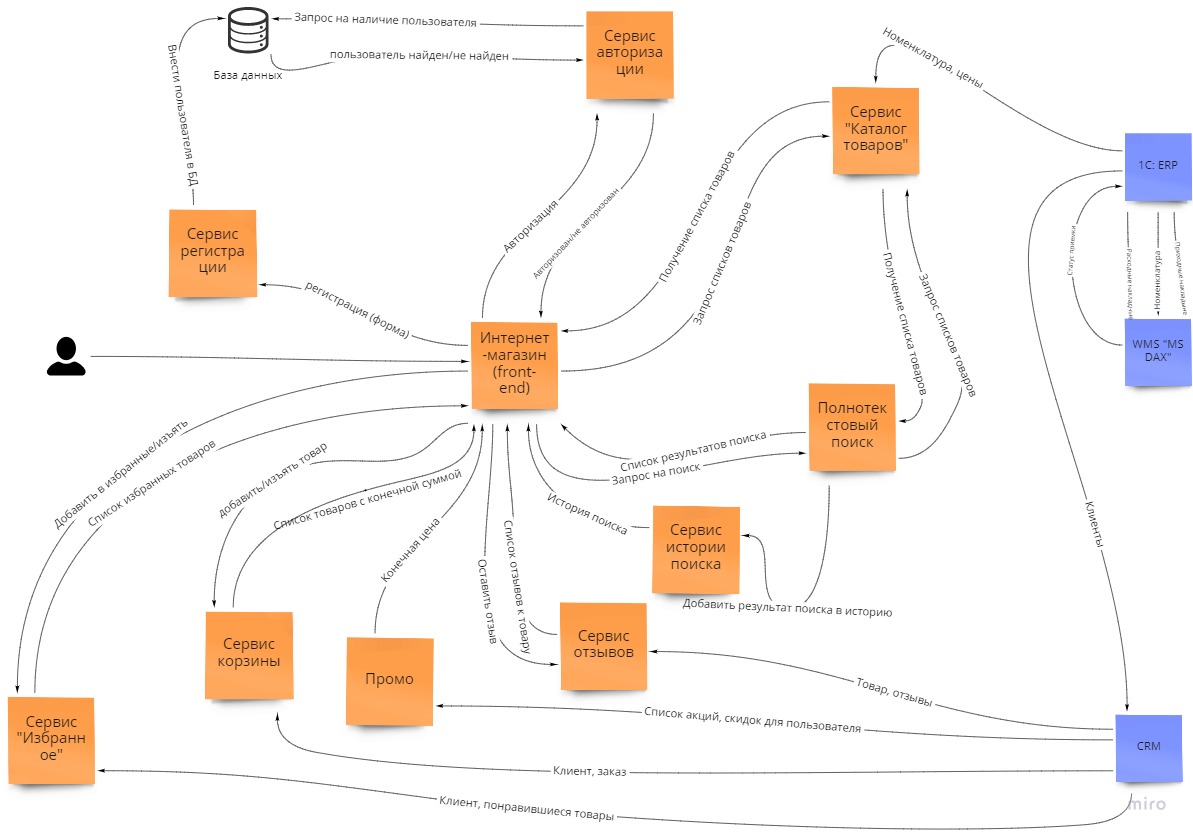
Необязательно. По материалам уроков БА.

|  |  |
| --- | --- |
| **Номер** | **Описание** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Описание бизнес-процесса

Необязательно. По материалам уроков БА.

# Архитектура решения



# Сервис каталога продуктов

## Функциональные требования

1. Возможность просматривать общий список товаров
2. Возможность отсортировать товары по рейтингу, количеству продаж, цене, бренду, названию, наличию
3. Возможность просматривать старую цену товара

## Нефункциональные требования

1. Каталог должен быть способен вмещать не менее 20 000 различных товаров
2. У товаров, которых нет в наличии, должна указываться старая цена
3. Сервис должен обслуживать не менее 1000 пользователей одновременно
4. Система должна быть горизонтально масштабируемой для увеличения числа одновременных пользователей

## Структура данных

Изображение выглядит как стол

Автоматически созданное описание

## Алгоритмы

### Каталог

### Поиск по автору

## Стек используемых технологий

# Сервис корзины

## Функциональные требования

1. Добавление товаров в корзину
2. Удаление товаров из корзины
3. Просмотр товаров в корзине
4. Просмотр общего количества товаров и общей стоимости товаров в корзине

## Нефункциональные требования

1. Возможность работы с корзиной как для авторизованных пользователей, так и для гостей
2. Корзина одного пользователя должна вмещать не менее 100 товаров
3. Товары в корзине гостей должны храниться в течение 10 дней
4. Товары в корзине авторизованных пользователей должны храниться месяц

## Структура данных

Изображение выглядит как стол

Автоматически созданное описание

## Алгоритмы

## Стек используемых технологий

# Сервис авторизации

## Функциональные требования

## Нефункциональные требования

## Структура данных

Изображение выглядит как стол

Автоматически созданное описание

## Алгоритмы

## Стек используемых технологий

# Frontend

Необязательно.

## Функциональные требования

## Нефункциональные требования

## Структура данных

## Алгоритмы

## Экранные формы

### Форма входа

### Каталог

### Корзина

### 

## Стек используемых технологий

# Интеграция

## Номенклатура (1С УПП ➡ Сервис каталога)

### Соответствие полей

### Триггеры запуска

.

### Алгоритмы

### Спецификация OpenAPI (Swagger)

## Клиенты (Сервис корзины ➡ CRM)

### Соответствие полей

### Триггеры запуска

### Алгоритмы

### Спецификация OpenAPI (Swagger)

## Каталог (Сервис каталога ➡ Frontend)

### Соответствие полей

### Триггеры запуска

### Алгоритмы

### Спецификация OpenAPI (Swagger)

# Ролевая модель

## Роли

1. Администратор
2. Гость
3. Авторизованный пользователь
4. Менеджер магазина

## Матрица доступа



# Система логирования

## Уровни логирования

* **Trace (DEV, TEST)** — вывод всего подряд. На тот случай, если Debug не позволяет локализовать ошибку.
* **Debug (DEV, TEST)** — журналирование моментов вызова «крупных» операций.
* **Info (PROD)** — разовые операции, которые повторяются крайне редко, но не регулярно.
* **Warning (STAGE)** — все, что может свидетельствовать о не штатном использовании.
* **Error (DAILY)** — ошибки системы.
* **Fatal (DAILY)** — критические ошибки

## Состав логов

Логи ведутся пакетным способом. Поэтому создадим 2 таблицы состава логов: таблицу с информацией по пакету и таблицу с данными пакета.

LOG\_PACKET

1. Уникальный идентификатор
2. Дата записи
3. Автор записи (ид юзера)
4. Признак обработки (0 или 1)
5. Тип записи (insert, update, delete)
6. Номер таблицы (название таблицы из которой пишем логи)
7. Номер таблицы с данными
8. Идентификатор записи пакета
9. Количество попыток обработки
10. Дата последней обработки

LOG

1. Уникальный идентификатор
2. Блоб поле для данных OLD
3. Блоб поле для данных NEW
4. Идентификатор записи пакета

Номер поля таблицы (название поля таблицы)

## Схема хранения логов (в днях)

Изображение выглядит как электроника, клавиатура

Автоматически созданное описание

# Система мониторинга

Изображение выглядит как текст

Автоматически созданное описание

# Стратегия резервного копирования

