

Интернет-магазин

Дизайн решения

Подготовлено: Адлер И.А.

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**](#_63cr9nctp4ep) **7**

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

[Функциональные требования](#_cmnaysb8jwdr) 8

[Нефункциональные требования](#_hyykwr4b2k95) 8

[Структура данных](#_3y06hkbl6l4k) 8

[Алгоритмы](#_nhskkro06tmi) 10

[Каталог](#_zai9eojzapd5) 10

[Поиск по автору](#_ooqukwmu32gq) 10

[Стек используемых технологий](#_a6jxcq7nqtcj) 10

[**Сервис корзины**](#_79ck44v3q8tc) **11**

[Функциональные требования](#_2xzhzoolvq76) 11

[Нефункциональные требования](#_g3gqpajg92ok) 11

[Структура данных](#_l99p2lexzk7j) 12

[Алгоритмы](#_8csxjulprb9e) 13

[Стек используемых технологий](#_lpgfk8samvb1) 14

[**Сервис авторизации**](#_sn7i14i7uqvt) **14**

[Функциональные требования](#_o1sdqyboqn57) 14

[Нефункциональные требования](#_q8by62wh2z59) 14

[Структура данных](#_8c5dp1tr7xwu) 14

[Алгоритмы](#_2epxqdurzcf) 14

[Стек используемых технологий](#_ib7er1rfk0hb) 14

[**Frontend**](#_gw6ncqprb49a) **15**

[Функциональные требования](#_5k6y0xpq8e7g) 15

[Нефункциональные требования](#_b47eprba7flv) 15

[Структура данных](#_1rui2bgqc843) 15

[Алгоритмы](#_rrkelykubz31) 15

[Экранные формы](#_kqqn4obgepx7) 15

[Форма входа](#_yro4izw9ved3) 15

[Каталог](#_lfd9ne3blaaw) 16

[Корзина](#_m7e75stsksss) 16

[Стек используемых технологий](#_wptolvap4vrd) 16

[**Интеграция**](#_3lqul3g5z5ja) **17**

[Номенклатура (1С УПП ➡ Сервис каталога)](#_5qxiubrayz15) 17

[Соответствие полей](#_7p36oq1uipue) 17

[Триггеры запуска](#_r9wjk0qdnzar) 17

[Алгоритмы](#_bolwwgxflvz2) 17

[Спецификация OpenAPI (Swagger)](#_83uqn4u6y6tz) 17

[Клиенты (Сервис корзины ➡ CRM)](#_165s0e2jxcm4) 20

[Соответствие полей](#_1oq3l2f2rxj0) 20

[Триггеры запуска](#_9g7weymrp60t) 21

[Алгоритмы](#_95spjw4i3jzr) 21

[Спецификация OpenAPI (Swagger)](#_admjmenenroz) 21

[Каталог (Сервис каталога ➡ Frontend)](#_u5ko13b8a07t) 23

[Соответствие полей](#_mk5y39wcg2gp) 23

[Триггеры запуска](#_mh0oicw7tozm) 24

[Алгоритмы](#_b4h4lb0zo3r) 24

[Спецификация OpenAPI (Swagger)](#_3fx8rvotmvev) 24

[**Ролевая модель**](#_gky9v2eq8wzl) **27**

[Роли](#_i641zxckhkpi) 27

[Матрица доступа](#_ltuzl4qf3ud4) 27

[**Система логирования**](#_xbzo22jtiwyq) **28**

[Уровни логирования](#_i9yy0mv0dnzl) 28

[Состав логов](#_a1caeiptxkdy) 28

[Схема хранения логов (в днях)](#_j0crfs30ip8g) 28

[**Система мониторинга**](#_877cbcsylc5t) **29**

[**Стратегия резервного копирования**](#_jas7832hxhmd) **30**

# Бизнес-цель

1. Увеличить продажи книг и сопутствующих товаров за счет широкого охвата аудитории через интернет.
2. Увеличить продажи книг и сопутствующих товаров за счет маркетинговых мероприятий на сайте через рекомендации и акции.
3. Упростить процедуру поиска нужного товара.
4. Упростить процедуру выбора товара.
5. Упростить процедуру оплаты и доставки товаров.

# Бизнес-требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Покупатель должен иметь возможность зарегистрироваться в личном кабинете разными вариантами на выбор: телефон, эл. почта, аккаунт google, yandex или соцсети (vk, fb, одноклассники). |
| 2. | Вход в личный кабинет не должен быть обязательным для покупки товаров. |
| 3. | Сайт перед завершением оформления товара должен предлагать покупателю войти в ЛК, чтобы сохранилась история его покупок. |
| 4. | При поиске книг по ключевым словам сайт должен выдавать поисковые результаты в порядке максимального совпадения по заданным словам. |
| 5. | Поисковый результат должен содержать фотографию товара и краткое описание. |
| 6. | Строка поискового результата должна быть кликабельной и переносить покупателя в карточку товара. |
| 7. | Карточка товара должна содержать несколько фотографий товара (от 2 до 7), описание товара и стоимость. |
| 8. | Пользователь должен иметь возможность добавить товар в корзину как из карточки товара, так и из поискового результата. |
| 9. | Под карточкой товара должна размещаться карусель с рекомендованными к покупке другими товарами (“с этим товаром часто покупают…”) |
| 10. | На главной странице сайта должна быть строка с текущими акциями и скидками. |
| 11. | Сайт должен отражать реальное количество товара на складе. |
| 12. | При отсутствии товара в наличии система должна рекомендовать замену. |

# Описание бизнес-процесса

*ссылка на bpmn-диаграмму:*

[BPMN-диаграмма Интернет-магазина Издательский Дом.png](https://drive.google.com/file/d/1lUuTlWLM3g7kT7_dNhjrEi-ThHIXNTEO/view)

Описание диаграммы:

Клиент попадает на главную страницу сайта, далее он либо пользуется каталогом на сайте, либо ищет нужную книгу через поисковую строку (исключающий шлюз XOR), поиск обращается к базе данных и получает поисковый результат в виде сформированного списка книг. Клиент ознакамливается с документом (результат поискового запроса). Далее идут взаимоисключающие события: 1) клиент не удовлетворен результатом и может многократно возвращаться на главную страницу для нового поиска, 2) клиент может покинуть сайт, 3) клиент добавляет понравившуюся книгу в корзину.

Для оформления заказа клиент обязан авторизоваться. После завершения добавления товаров в корзину клиент либо “бросает” ее и уходит, либо авторизовывается в системе. Если клиент регистрируется впервые, его данные попадают в CRM. Если клиент уже зарегистрирован, то CRM проверяет подлинность введенных данных и разрешает оформление заказа.

Клиент оформляет заказ, затем оплачивает его. В случае ошибки оплаты возникает прерывающее событие с описанием ошибки и происходит переадрессация на страницу оплаты для повторной попытки. После успешной оплаты информация об этом передается на сайт. Сайт получает сообщение об оплате заказа и делает два обязательных действия (шлюз AND): отправляет клиенту на почту детали заказа и подтверждение оплаты, а также ставит задачу сборщикам товара на сбор данного заказа.

После положенного на сбор времени заказ передается в отдел доставки (свёрнутый пул), клиент получает заказ, процесс завершен.

# 

# Архитектура решения

| Центральным блоком является Frontend, т.к. это первое, что видит клиент. Frontend получает информацию из двух ресурсов: собственно каталог продукции, а также рекомендации к покупкам и акции (т.е. каталоги передают информацию на главный экран).  Покупатель по желанию может пройти авторизацию, тогда его данные попадут в CRM-систему. Покупатель выбирает понравившийся товар и переходит в карточку товара. В карточку товара информация о количестве и ценах попадает из каталога складского учета (Склад). Также в карточку товара поступает информация из Каталога акций и рекомендаций (“С этим товаром покупают…”, а при отсутствии товара - его аналог/замена). Из карточки товары попадают в корзину (сервис Заказа).  Сформированный заказ рассчитывается и окончательная сумма передается в Сервис оплаты, который обратно возвращает информацию об успешной/неуспешной транзакции. После подтверждения оплаты корзина передает на Склад команду об изменении количества оставшегося товара. А информация о состоянии заказа передается в CRM. CRM в свою очередь может передавать в Каталог рекомендаций новые рекомендации (в т.ч. персональные для этого покупателя в дальнейшем), а также может передать информацию в Сервис рассылки, чтобы простимулировать ушедшего покупателя вернуться и завершить покупку.  В случае отмены покупки пользователь авторизуется на сайте, CRM система хранит данные о его заказе, он переходит в Сервис заказа (история заказов) и через Систему оплаты аннулирует покупку и оставляет заявление на возврат денежных средств. Информация об аннулировании Заказа передается в заказ/корзину и потом в CRM. Корзина передает информацию на склад об увеличении товарных единиц.  Двусторонние стрелки на диаграмме свидетельствуют о том, что обмен информацией происходит в обе стороны. | |
| --- | --- |
|  | |

# Сервис каталога продуктов

## Функциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Каталог должен отображаться на главной странице в виде сгруппированного по жанрам списка |
| 2. | По каталогу должен работать фильтр по авторам, жанру, издательству, интервалам цен, по акциям. |
| 3. | Результат поискового запроса в случае наличия одной и той же книги разного года издания должен быть отсортирован по более поздней дате издания |
| 4. | Поисковый результат должен отображаться в виде вертикального списка и содержать главную фотографию книги, название, имя автора, цену, 3 строки с описанием товара, заканчивающиеся многоточием и кнопку быстрого добавления в корзину |
| 5. | Редактировать, добавлять, удалять товары из каталога должны пользователи, имеющие соответствующее разрешение на эти действия. |
| 6. | Данные о действиями с товарами в каталоге, а также время этих действий должны хранится в специальных логах, доступ к которым должен быть только у администратора системы |

## Нефункциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Поиск по каталогу должен осуществляться не более чем за 15 секунд. |
| 2. | Объем используемой оперативной памяти при поиске не должен превышать 256 Мб |

## Структура данных

| **1. Концептуальный уровень** | |
| --- | --- |
|  | |
| Данные собственно в каталог попадают из таблицы с книгами, находящейся в базе данных MySQL “shop”. Таблица с книгами связана с таблицей Склад, где содержится информация о месте книги на стеллаже, а также с таблицей жанров.  Каждая книга должна иметь одно и только одно место на складе (связь “один к одному”), в то время как склад может хранить много книг или быть пустой (связь “от нуля ко многим”).  У каждой книги для удобства фильтрации и группировки будет только один жанр (“один к одному”), один жанр может описывать много книг (связь “от нуля ко многим”). | |
| **2.** **Логический уровень** | |
|  | |
| **3.** **Физический уровень** | |
|  | |
| Каждая книга имеет уникальный bookID, он же будет являться Primary Key в таблице Склад. В таблице жанров у каждого жанра свой уникальный серийный genreID, он будет являться Foreign Key в таблице книг. | |

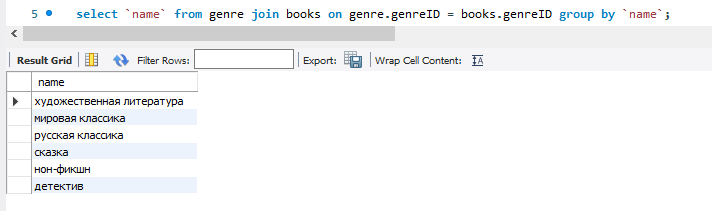
## 

## Алгоритмы

### **Каталог**

В функциональных требованиях было требование для каталога - группировка по жанрам.

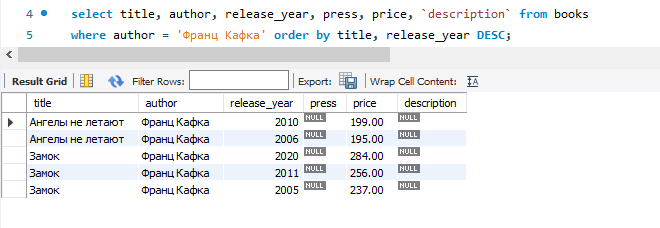
Через join объединяем таблицу с книгами и таблицу с жанрами, группируем по наименованию жанра:



### **Поиск по автору**

Одним из требований (в случае наличия книг с одинаковым названием, но отличающихся годами издания) было формирования списка поискового результата от более позднего года издания к более раннему. Кроме того, необходимо учитывать, что один автор может написать много разных книг.

Реализуем поиск по автору с сортировкой по названию и году издания от более свежего по убыванию:



## Стек используемых технологий

* MySQL
* C#

# Сервис корзины

## Функциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | При добавлении товара в корзину сайт не должен сразу открывать окно корзины, чтобы пользователь продолжал выбирать товары из каталога. |
| 2. | В случае если покупатель добавляет товары в корзину, но потом не оплачивает заказ и уходит, заказу должен присваиваться статус “не оплачен” и соответствующая информация должна передаваться в CRM для последующей работы с покупателем. |
| 3. | В случае если покупатель добавляет товары в корзину, а потом очищает ее и уходит, заказу должен присваиваться статус “пустой” и соответствующая информация должна передаваться в CRM. |
| 4. | Кнопка “оформить заказ” должна становиться активной в корзине, если в корзине есть хотя бы 1 товар. И должна быть неактивной в пустой корзине. |
| 5. | Платежная система после успешной транзакции должна передавать в корзину информацию об оплате, после чего заказ должен приобретать статус “оплачен”, а соответствующая информация об этом должна быть передана в CRM. |
| 6. | После успешной оплаты заказа информация должна быть передана на склад для изменения количества товара. |

## Нефункциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Реализовать базу данных в noSQL для упрощения работы с заказом |
| 2. | При добавлении товара в корзину товар должен отображаться через 0.1 секунду после нажатия кнопки “добавить в корзину”. |

## 

## 

## Структура данных

| **1. Концептуальный уровень** | |
| --- | --- |
|  | |
| Заказ содержит книги. Один заказ может содержать много книг или быть пустым (пользователь добавил книги в заказ, но все очистил и ушел), связь “от нуля ко многим”. Каждая книга может быть добавлена в разные заказы, либо может оставаться невостребованной, связь “от нуля ко многим”.  Заказ создается пользователем. Пользователь может ни создать ни одного заказа (ничего не добавит в корзину), либо создать много заказов, связь “от нуля ко многим”. Созданный заказ должен иметь одного и только одного создателя, связь “один к одному”. | |
| **2.** **Логический уровень** | |
|  | |
| Чтобы избежать связей “многие ко многим” между таблицами, вводим дополнительную таблицу Заказ-товар. Созданная пара заказ-товар должна иметь только один id заказа и только один id книги, связи “один к одному”. | |
| **3.** **Физический уровень** | |
|  | |

## Алгоритмы

Поиск всех незакрытых заказов:

по id статуса заказов, соответствующий наименованию”новый”, находим все незакрытые заказы. Для поиска незакрытых более недели заказов делаем дополнительное условие по дате.

Определение суммы всех заказов одного покупателя:

По id покупателя из таблицы заказов делаем выборку за всё время, суммируем все суммы заказов. Результат - общая сумма всех заказов. На основе этих данных можно присваивать покупателям разные статусы: серебряный, золотой и т.д.

## Стек используемых технологий

* MongoDB

# Сервис авторизации

## Функциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Система должна перенаправлять пользователя на страницу авторизации после добавления товара в корзину и нажатия кнопки “оформить заказ”. Без авторизации пользователь не может оформить и оплатить заказ. |
| 2. | Система должна поддерживать возможность авторизации через социальные сети и через google-аккаунт |

## Нефункциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Реализовать базу данных пользователей, объектов, типов пользователей и прав доступа в MySQL |

## 

## Структура данных

| **1. Концептуальный уровень** | |
| --- | --- |
|  | |
| Каждый пользователь имеет определенную роль (только одну). Каждая роль имеет по крайней мере одно право. Права распространяются на объекты. | |

## 

## 

| **2.** **Логический уровень** | |
| --- | --- |
|  | |
| Чтобы избежать связи “многие ко многим” вводим таблицу Роль-объект-право. | |
| **3.** **Физический уровень** | |
|  | |

## Алгоритмы

а) Аутентификация по логину-паролю.

1. Входные параметры: логин + пароль;
2. Если ок, возвращаем 200;
3. Если ошибка, возвращаем 401.

б) При поступлении запроса в метод API авторизации:

1. По входному параметру логин найти запись в таблице Покупатели (customers), поле ID;
2. Если нашли запись, возвращаем 200, иначе возвращаем 401;
3. По входному параметру объект ищем запись в таблице Объекты (objects), поле ID;
4. Если не нашли, возвращаем 404;
5. Проверяем в таблице Права доступа наличие записи, соответствующей запрошенному пользователем объекту и праву;
6. Если не нашли, возвращаем 403.

## 

## Стек используемых технологий

* MySQL

Frontend

## Функциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Реализовать возможность быстрого добавления товаров в корзину из поискового результата |
| 2. | Навигация сайта должна включать следующие разделы: жанры, новинки, акции, строку поиска, личный кабинет, корзину, уведомления и избранное, а также ссылку на информацию о юр. лице и контактах организации. |
| 3. | На главной странице должны отображаться новинки и акции магазина |
| 4. | Новинки на главной странице должны содержать фотографию, название, автора, краткое описание. |
| 5. | Новинки должны умещаться в одну строку и прокурчиваться вправо и влево по принципу карусели |

## Нефункциональные требования

| **Номер** | **Описание** |
| --- | --- |
| 1. | Скорость ответа сайта на запрос должна составлять не более 2 сек. и должна быть одинаковой при работе с различных устройств (ПК, планшет, телефон) |
| 2. | Работа в современных браузерах Internet Explorer 5.5 и выше, Mozilla Firefox 7.01 и выше, Safari 5.1.7 и выше, Google Chrome 13.0.772.0 (быстрая загрузка страниц). |
| 3. | Используемый язык программирования С# |
| 4. | Должна быть предусмотрена возможность для авторизации администратора сайта и менеджерам по доставке и по работе с клиентами. |
| 5. | Не хранить пароли БД в открытом виде |
| 6. | Система должна быть устойчива к одновременным многочисленным регистрациям на сайте (более 100) |

## Структура данных

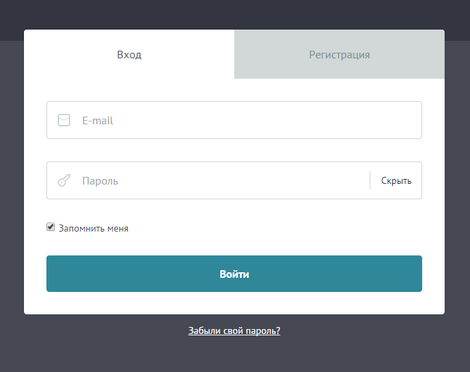
Frontend не имеет собственной базы данных и получает данные из БД каталога и CRM системы.

## Алгоритмы

Frontend получает данные из баз данных (каталога, CRM, заказов и т.д.) с помощью REST-запросов и языка программирования

## Экранные формы

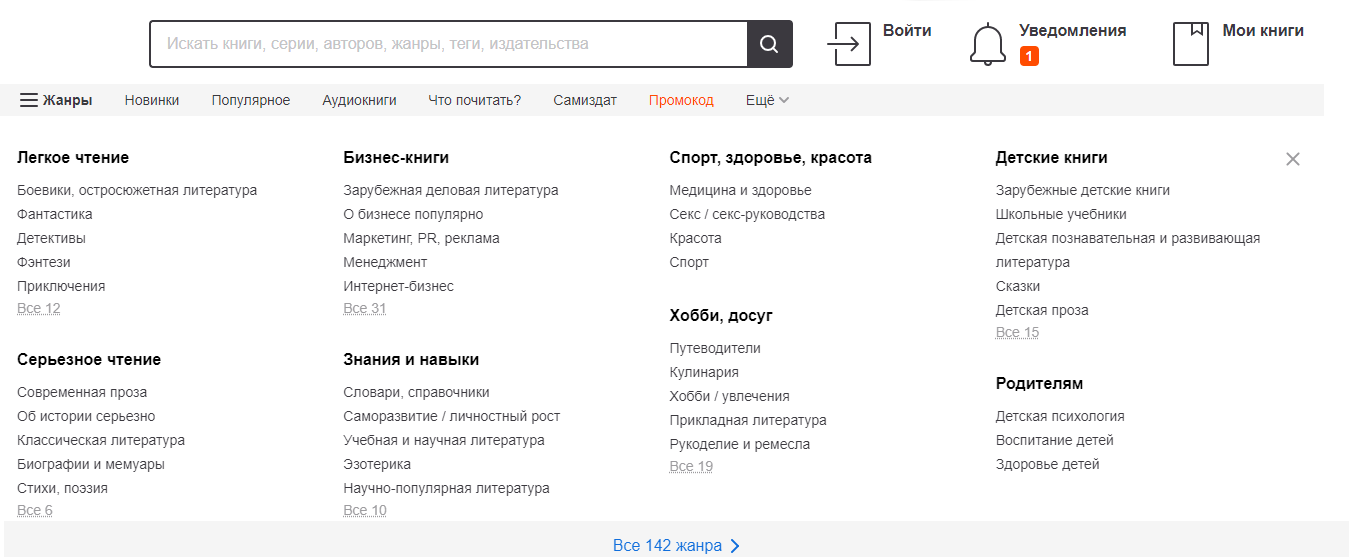
### Форма входа



### 

### 

### Каталог



### 

### Корзина

### 

## Стек используемых технологий

* UI/UX Design
* C#

# Интеграция

## Номенклатура (1С УПП ➡ Сервис каталога)

### Соответствие полей

| **наименование параметра** | **формат 1С** | **преобразование в формат каталога** | **комментарий** |
| --- | --- | --- | --- |
| id | 10000001-001 | 1 | серийный порядковый номер |
| название книги | “Название книги” | Название книги | без кавычек |
| автор | Иванов Александр Александрович | Александр Иванов | сначала имя, потом фамилия (без отчества) |
| жанр | детектив | детектив | преобразование отсутствует |
| издательство | АО “Дрофа” | Дрофа | без кавычек и формы образования юр. лица |
| год | 2000 | 2000 | преобразование отсутствует |

### Триггеры запуска

1. По нажатию кнопки в 1С “синхронизировать данные с сайтом”.
2. Автоматическое обновление раз в сутки в 23.00 ч.

### Алгоритмы

При нажатии в 1С кнопки “синхронизировать с сайтом” происходит обновление данных товаров (put), добавление новых товаров (post) и удаление в каталоге товаров, отсутствующих в 1С (delete).

### Спецификация OpenAPI (Swagger)

swagger: "2.0"

info:

description: "Nomenclature from 1C to a catalog of a bookstore."

version: "1.0.0"

title: "Nomenclature"

contact:

email: "apiteam@books.ru"

license:

name: "Apache 2.0"

url: "http://www.apache.org/licenses/LICENSE-2.0.html"

host: "books.ru"

basePath: "/v1"

**paths:**

**/books:**

**put:**

tags:

- "book"

summary: "Put books"

description: "Update a book"

operationId: "putBooks"

consumes:

- "application/json"

- "application/xml"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "query"

name: "bookID"

description: "book id filter to put a book"

required: true

type: "integer"

responses:

"200":

description: "Successful"

schema:

type: "array"

items:

$ref: "#/definitions/Book"

"400":

description: "Error"

security:

- api\_key: []

**post**:

tags:

- "book"

summary: "Creatung a new book"

description: "Creating a new book"

operationId: "createBooks"

consumes:

- "application/json"

- "application/xml"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "body"

name: "bookID"

description: "New book"

required: true

schema:

$ref: "#/definitions/Book"

responses:

"200":

description: "Successful"

schema:

type: array

items:

$ref: "#/definitions/Book"

"400":

description: "Error"

security:

- api\_key: []

**delete**:

tags:

- "book"

summary: "Delete books"

description: "Delete a book"

operationId: "deleteBooks"

consumes:

- "application/json"

- "application/xml"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "query"

name: "bookID"

description: "Book id to delete the same"

required: true

type: "integer"

responses:

"200":

description: "Successful"

schema:

type: "array"

items:

$ref: "#/definitions/Book"

"400":

description: "Error"

security:

- api\_key: []

securityDefinitions:

api\_key:

type: "apiKey"

name: "api\_key"

in: "header"

definitions:

Book:

type: "object"

properties:

id:

type: "integer"

format: "int64"

name:

type: "string"

author:

type: "string"

price:

type: "number"

xml:

name: "Book"

ApiResponse:

type: "object"

properties:

code:

type: "integer"

format: "int32"

type:

type: "string"

message:

type: "string"

externalDocs:

description: "Find out more about Swagger"

url: "http://swagger.io"

## Клиенты (Сервис корзины ➡ CRM)

### Соответствие полей

| **наименование параметра** | **формат корзины** | **преобразование в формат СRM** | **комментарий** |
| --- | --- | --- | --- |
| order id | 1 | 0001 |  |
| название книги | Название | Название |  |
| автор | Александр Иванов | Александр Иванов |  |
| цена | 111,11 | 111.11 | преобразование разделителя |
| дата заказа | 01.01.2020 | 01.01.2020 12:00 | добавление времени |
| user id | 1 | 10-0001 |  |

### Триггеры запуска

1. По нажатию в корзине кнопки “оформить заказ”.
2. По нажатию в личном кабинете кнопки “история заказов”.

### Алгоритмы

После добавления товаров в корзину, при нажатии кнопки “оформить заказ” происходит добавление заказа в CRM (post) со статусом “new”.

При нажатии кнопки в личном кабинете “история заказов” подгружаются заказы данного клиента (get).

### Спецификация OpenAPI (Swagger)

swagger: "2.0"

info:

description: "Service of orders for a bookstore."

version: "1.0.0"

title: "Basket"

contact:

email: "apiteam@books.ru"

license:

name: "Apache 2.0"

url: "http://www.apache.org/licenses/LICENSE-2.0.html"

host: "books.ru"

basePath: "/v1"

**paths:**

**/orders:**

**get:**

tags:

- "order"

summary: "Get orders"

description: "Get list of orders"

operationId: "getOrders"

consumes:

- "application/json"

- "application/xml"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "query"

name: "userID"

description: "User id for filter to get orders"

required: true

type: "integer"

responses:

"200":

description: "Successful"

schema:

type: "array"

items:

$ref: "#/definitions/Order"

"400":

description: "Error"

security:

- api\_key: []

**post:**

tags:

- "order"

summary: "Update orders"

description: "Update existing orders or create new"

operationId: "updateOrders"

consumes:

- "application/json"

- "application/xml"

produces:

- "application/xml"

- "application/json"

parameters:

- in: "body"

name: "order\_number"

description: "New order"

required: true

schema:

$ref: "#/definitions/Order"

responses:

"200":

description: "Successful"

schema:

type: array

items:

$ref: "#/definitions/Order"

"400":

description: "Error"

security:

- api\_key: []

securityDefinitions:

api\_key:

type: "apiKey"

name: "api\_key"

in: "header"

definitions:

Order:

type: "object"

properties:

id:

type: "integer"

format: "int64"

orderId:

type: "integer"

format: "int64"

quantity:

type: "integer"

format: "int32"

shipDate:

type: "string"

format: "date-time"

status:

type: "string"

description: "Order Status"

enum:

- "new"

- "cancelled"

- "delivered"

complete:

type: "boolean"

default: false

xml:

name: "Order"

ApiResponse:

type: "object"

properties:

code:

type: "integer"

format: "int32"

type:

type: "string"

message:

type: "string"

externalDocs:

description: "Find out more about Swagger"

url: "http://swagger.io"

## Каталог (Сервис каталога ➡ Frontend)

### Соответствие полей

| **наименование параметра** | **формат каталога** | **преобразование в формат Frontend** | **комментарий** |
| --- | --- | --- | --- |
| id | 1 | - | id на сайте не указывается |
| название книги | Название книги | Название книги | преобраз. отсут. |
| автор | Александр Иванов | Александр Иванов | преобраз. отсут. |
| жанр | детектив | детектив | преобраз. отсут. |
| издательство | Дрофа | Дрофа | преобраз. отсут. |
| год издания | 2000 | 2000 | преобраз. отсут. |

### Триггеры запуска

1. При выборе на сайте жанра из выпадающего списка каталога.
2. При поиске книги по ключевым словам через строку поиска.
3. Обновление на главной странице сайта раздела новых книг происходит два раза в сутки в 12 ч и в 20 ч, а также по нажатию кнопки “обновить информацию на сайте”

### Алгоритмы

При выборе жанра либо поиске книги через строку поиска Frontend обращается к каталогу за списком книг.

Два раза в сутки в установленное время каталог размещает на frontend новые добавленные книги или обновляет информацию в существующих книгах, либо удаляет неактуальные товарные единицы с сайта.

### Спецификация OpenAPI (Swagger)

swagger: "2.0"

info:

description: "Catalog for a bookstore."

version: "1.0.0"

title: "Catalog"

contact:

email: "dev@books.ru"

license:

name: "Apache 2.0"

url: "http://www.apache.org/licenses/LICENSE-2.0.html"

host: "books.ru"

basePath: "/v1"

paths:

**/catalog:**

**get:**

tags:

- "catalog"

summary: "getCatalog"

description: "Get list of books"

operationId: "getCatalog"

consumes:

- "application/json"

produces:

- "application/json"

parameters:

- name: "author"

in: "query"

description: "Author for filter"

required: false

type: "string"

responses:

"200":

description: "Successful"

schema:

type: "array"

items:

$ref: "#/definitions/Book"

"400":

description: "Error"

security:

- api\_key: []

**/book:**

**post:**

tags:

- "catalog"

summary: "update books"

description: "Update existing boor or create new one."

operationId: "updateBook"

consumes:

- "application/json"

produces:

- "application/json"

parameters:

- name: "book"

in: "body"

description: "New book"

required: true

schema:

$ref: "#/definitions/Book"

responses:

"200":

description: "Successful"

schema:

type: "array"

items:

$ref: "#/definitions/Book"

"400":

description: "Error"

security:

- api\_key: []

**delete:**

tags:

- "catalog"

summary: "delete books"

description: "Delete books from frontend."

operationId: "deleteBook"

consumes:

- "application/json"

produces:

- "application/json"

parameters:

- name: "book"

in: "body"

description: "Book for deletion"

required: true

schema:

$ref: "#/definitions/Book"

responses:

"200":

description: "Successful"

schema:

type: "array"

items:

$ref: "#/definitions/Book"

"400":

description: "Error"

security:

- api\_key: []

securityDefinitions:

api\_key:

type: "apiKey"

name: "api\_key"

in: "header"

definitions:

Book:

type: "object"

properties:

id:

type: "integer"

name:

type: "string"

url:

type: "string"

author:

type: "string"

price:

type: "number"

externalDocs:

description: "Find out more about Swagger"

url: "http://swagger.io"

# Ролевая модель

## Роли

| **№** | **Роль** | **Описание** |
| --- | --- | --- |
| 1 | Гость | Неавторизованный пользователь |
| 2 | Клиент | Авторизованный пользователь |
| 3 | Администратор сайта |  |
| 4 | Менеджер клиентской поддержки |  |
| 5 | Менеджер доставки |  |

## Матрица доступа

|  | Каталог | Карточка товара | Корзина/заказ | Отзыв | Информация о клиенте |
| --- | --- | --- | --- | --- | --- |
| Гость | R | R | C R | R | - |
| Клиент | R | R | C\* R\* U\* D\* | C\* R U\* D\* | C\* R\* U\* D\* |
| Менеджер по доставке | R | R | R U | R |  |
| Администратор сайта | C R U D | C R U D | C R U D | C R U D | C R U D |
| Менеджер клиентской поддержки | R U | R U | C R U D | R U | R U |

\*только свой

Примечание:

C - create

R - read

U - update

D - delete

# Система логирования

## Уровни логирования

|  | DEV | TEST | STAGE | PROD | DAILY |
| --- | --- | --- | --- | --- | --- |
| debug | **𐠨** | **𐠨** |  |  | **𐠨** |
| info | **𐠨** | **𐠨** | **𐠨** |  | **𐠨** |
| warning | **𐠨** | **𐠨** | **𐠨** |  | **𐠨** |
| error | **𐠨** | **𐠨** | **𐠨** | **𐠨** | **𐠨** |
| fatal | **𐠨** | **𐠨** | **𐠨** | **𐠨** | **𐠨** |

Комментарии: для разработки и тестирования будем записывать все уровни логов. Для среды подготовки релиза достаточно уровня info, для продакшена - уровня ошибок.

Для среды daily будем записывать все уровни, т.к. в ней можно тестировать ПО на реальных данных, но хранить логи будем один день (протестировали - посмотрели логи, на следующий день очистили).

## Состав логов

* дата и время записи,
* тип лога (info, error и т.д.),
* место в коде,
* описание (действие, что произошло),
* координаты клиента, заказа (id).

## Схема хранения логов (в днях)

|  | DEV | TEST | STAGE | PROD | DAILY |
| --- | --- | --- | --- | --- | --- |
| дней хранения | **7** | 7 | 14 | 180 | 1 |

## 

# 

# Система мониторинга

1. **Метрики для сервисов**

| Метрики/Сервис | Frontend | Сервис авторизации | Сервис каталога | Сервис корзины |
| --- | --- | --- | --- | --- |
| Метрики фронта | - first contentful paint  - time to interactive  - время реакции на действия |  |  |  |
| Метрики инфраструктуры | - CPU usage  - RAM usage  - disk space usage | - CPU usage  - RAM usage  - disk space usage | - CPU usage  - RAM usage  - disk space usage | - CPU usage  - RAM usage  - disk space usage |
| Метрики БД |  | - CPU usage  - RAM usage  - disk space usage  - время отклика  - кол-во подключений | - CPU usage  - RAM usage  - disk space usage  - время отклика  - кол-во подключений | - CPU usage  - RAM usage  - disk space usage  - время отклика  - кол-во подключений |
| Метрики API |  | - 4xx rate  - 5xx rate  - mediana  - p95  - health is OK | - 4xx rate  - 5xx rate  - mediana  - p95  - health is OK | - 4xx rate  - 5xx rate  - mediana  - p95  - health is OK |

1. **Показатели мониторинга объектов сервисов**

| Объект/Сервис | Frontend | Сервис авторизации | Сервис каталога | Сервис корзины |
| --- | --- | --- | --- | --- |
| Дисковое пространство | 10% | 5% | 15% | 10% |
| Процессор | 60% - warning  80% - critical | 60% - warning  80% - critical | 60% - warning  80% - critical | 60% - warning  80% - critical |
| Оперативная память | 75% | 75% | 75% | 75% |
| Select к БД | - | Запрос:  Время ответа: 2 сек  Кол-во коннектов: 500 | Запрос:  Время ответа: 2 сек  Кол-во коннектов: 500 | Запрос:  Время ответа: 1 сек  Кол-во коннектов: 500 |
| HTTP GET | <https://shop.books.ru/>  0,5 сек | <https://auth.books.ru/check/>  0,5 сек | <https://catalog.books.ru/check/>  1,5 сек | <https://cart.books.ru/check/>  1 сек |

# Стратегия резервного копирования

|  | DEV | TEST | STAGE | DAILY | PROD |
| --- | --- | --- | --- | --- | --- |
| Назначение | Разработка | Тестирование | Подготовка релиза | Копия прода для анализа | Промышленная эксплуатация |
| Обновление с ПРОДа | по запросу | по запросу | по запросу | ежедневно | - |
| Объекты для резерв. копир-я | Базы данных сервисов | Базы данных сервисов | Базы данных сервисов | Базы данных сервисов | Базы данных сервисов |
| Порядок резервного копир-я | 1) Полная резерв. копия - 1 раз / неделю (сб в 23.59)  2) Инкрементная копия ежедневно в 23.59 | Полная резервная копия - 1 раз / неделю (сб в 23.59) | Полная резервная копия - 1 раз / мес (30 числа в 23.59) | - | 1) Ежедневная полная копия ночью (вс-пт).  2) Еженедельная полная копия ночью с сб на вс.  3)Инкрементная копия каждые 2 часа с 8.00 до 21.00).  4) Резервная копия журнала транзакций БД каждые 15 мин (с 8.00 до 22.00) |
| Сроки хранения резерв. копий | 1) Еженедельная полная копия - 6 мес. (4\*6+2=26 копий)  2) Инкрементная копия - 2 недели (7\*2=14 копий) | Еженедельная полная резервная копия - 30 дн  (1\*4=4) | Полная копия - 6 мес. (1\*6 = 6 копий) | - | 1) Ежедневная полная копия - 3 мес. (3\*30=90 копий каждой базы)  2) Еженедельная полная копия - 6 мес. (4\*6+2=26 копий каждой БД)  3)Инкрементная копия - 1 неделя (7\*7=49 копий каждой БД)  4) Копия журнала транзакций - 2 суток (14\*4=56 копий журнала транзакций) |