

СОДЕРЖАНИЕ

	стр.
СОДЕРЖАНИЕ.....	1
ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....	2
ВВЕДЕНИЕ.....	2
1. ОБЗОРНАЯ ЧАСТЬ.....	5
1.1. Описание предметной области.....	5
1.1.1. Описание серверной архитектуры.....	5
1.1.2. Описание модели внедрения программных продуктов Agile Scrum.....	6
1.2. Исследование существующих технологий	6
1.2.1. Перечень функций, подлежащих автоматизации	6
1.2.2. Выбор и обоснование критериев качества	6
1.2.3. Анализ аналогов и прототипов.....	7
1.2.3.1. Swagger	8
1.2.3.2. API Blueprint.....	10
1.2.3.3. RAML	12
1.2.3.4. Ручной метод сопровождения API-документации	12
1.2.3.5. Postman.....	14
1.2.4. Сравнение аналогов и прототипов	16
1.2.5. Вывод.....	17
2. РАСЧЕТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ.....	18
2.1. Определение требований к системе.....	18
2.2. Разработка структуры автоматизированной системы	19
2.3. Разработка структуры интерфейса взаимодействия пользователя с системой	20
2.4. Разработка алгоритмов программных модулей	20
2.5. Разработка плана проведения тестирования	20
3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ.....	20
3.1. Реализация разработанных алгоритмов	20
3.2. Тестирование и отладка системы	21
3.3. Руководство пользователя	21
ЗАКЛЮЧЕНИЕ	21
СПИСОК ЛИТЕРАТУРЫ	21

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

- 1) REST API – это набор правил, по которым следует обращаться к серверу для отправки или получения данных.
- 2) Клиент – любое приложение которое делает запросы на сервер. Например, в роли клиента может выступать веб браузер, когда пользователь открывает веб-сайт
- 3) АС – автоматизированная система
- 4) ПО – Программное обеспечение
- 5) API-документация – это техническая документация, в которой фиксируются инструкции о том, как использовать программное API.
- 6) UI (User Interface) – пользовательский интерфейс
- 7) URL (Uniform Resource Locator) - адрес сайта или отдельной страницы в сети интернет
- 8) JSON (JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript.
- 9) ТТ (Template Toolkit) - perl-библиотека для работы с шаблонами, позволяющая разделять код, данные и представление.
- 10) БД (База данных) – это совокупность систематизированных особым образом данных, находящаяся в памяти вычислительной системы. Для работы с БД используются специальные средства – системы управления базами данных (СУБД).

ВВЕДЕНИЕ

На сегодняшний день большинство крупных IT компаний для взаимодействия сервера и клиента используют REST API [1].

Компании вроде Яндекса, Google и т.п. Предоставляют открытые API методы своих сервисов чтобы разработчики могли интегрироваться с ними.

Например, при получении данных о пользователе, информация о котором храниться в БД (базе данных) на сервере необходимо указать путь до сервера (URI), идентификатор пользователя (ID) и метод (Method) по которому сервер поймет, что нужно сделать с ресурсом, в данном случае вернуть информацию о пользователе. Данный процесс «общения» клиента и сервера, представлен на рисунке 1.1.

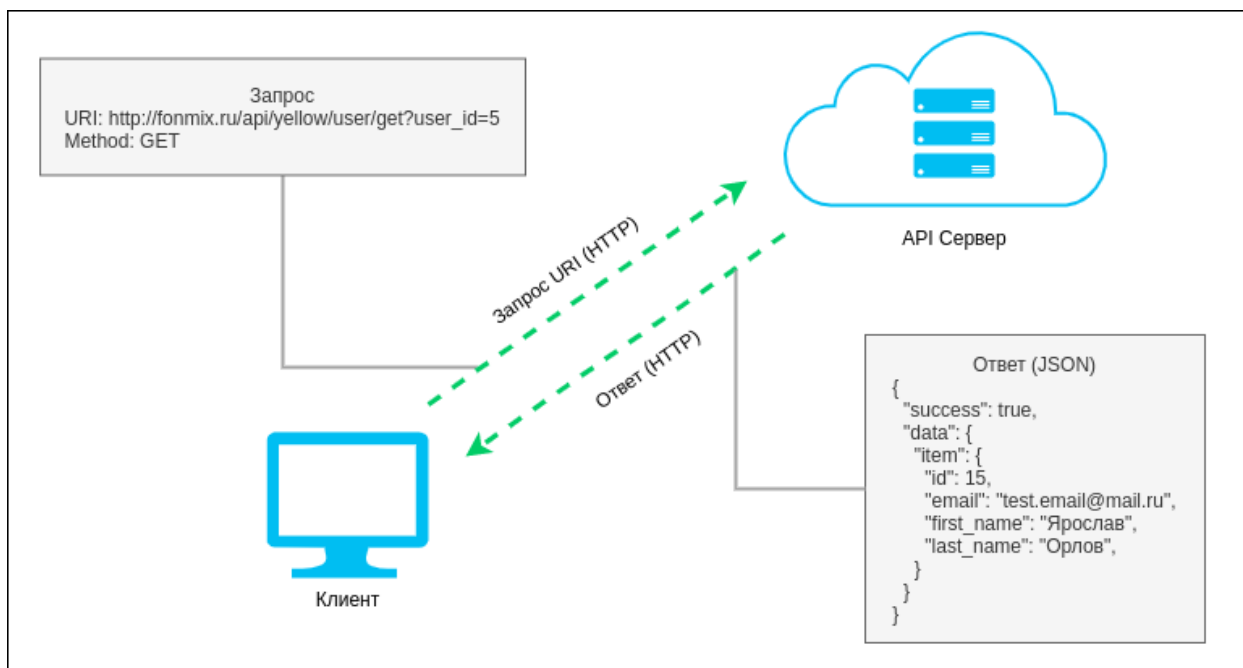


Рисунок 1.1 – Схема получения информации о пользователе

Помимо написания самих API методов необходимо написание подробной документации по ним, поскольку без нее попросту не удастся воспользоваться методом. А также не менее важно поддерживать документацию в актуальном состоянии поскольку если документация будет неправильная или устаревшая, то велика вероятность ошибок и в конечном итоге может сказываться на качестве и стоимости продуктов. Поэтому написание API-документации очень важная и актуальная тема.

API-документация представляет собой....

Что такое API-документация? Пример!

Компания ООО «ФорМакс» разрабатывает продукт Fonmix, серверная часть которого полностью базируется на технологии REST API, то есть взаимодействие любого пользователя с сервером Fonmix осуществляется через REST API.

Основными клиентами [3] для сервера Fonmix являются:

- 1) Веб-сайт fonmix.ru – представляет собой веб интерфейс, в котором пользователи [2] могут управлять музыкой в своих заведениях: создавать плейлисты, составлять музыкальное расписание, добавлять рекламу в перерывах между песнями и т.п.
- 2) FM.Player – кроссплатформенный медиа проигрыватель разрабатываемый также в компании ООО «ФорМакс», с помощью которого воспроизводится медиа контент правообладателей.
- 3) Правообладатель – это исполнитель и изготовитель фонограмм, с которым заключается договор о дистрибуции контента и предоставлении отчетов об использовании.

Тут нужно как-то подвести от того где я работаю к доке

Целью данной работы является создание системы автоматического сопровождения API-документации, позволяющей ускорить и повысить качество разработки. В соответствии с поставленной целью, работа над АС (автоматизированной системой) была разделена на несколько этапов, в рамках которых решались следующие задачи:

- анализ предметной области
- обзор и сравнение современных технологий по сопровождению API-документации
- выделение перечня функций, подлежащих автоматизации
-

1. ОБЗОРНАЯ ЧАСТЬ

1.1. Описание предметной области

1.1.1. Описание серверной архитектуры

Серверная часть проекта Fonmix на разделена на микросервисы.

Микросервисная архитектура¹ – вариант сервис-ориентированной архитектуры программного обеспечения, направленный на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей – микросервисов.

Основными микросервисами являются:

- FM.Core – Основной сервис для работы с клиентам. Количество API методов 253
- FM.CRM – Сервис для получения данных о пользователях для дальнейшего их анализа. Количество API методов 153
- FM.ID – Сервис для авторизации пользователей. Количество API методов 23
- FM.Notify – Сервис для отправки уведомлений пользователям. Количество API методов 34
- FM.Store – Сервис для хранения и обработки файлов пользователей. Количество API методов 15
- FM.Media – Сервис для хранения и распространения медиа контента правообладателей. Количество API методов 36

Итого, общее количество методов 514

Схема взаимодействия клиентов и сервера представлена на рисунке 1.2.

¹ https://ru.wikipedia.org/wiki/Микросервисная_архитектура

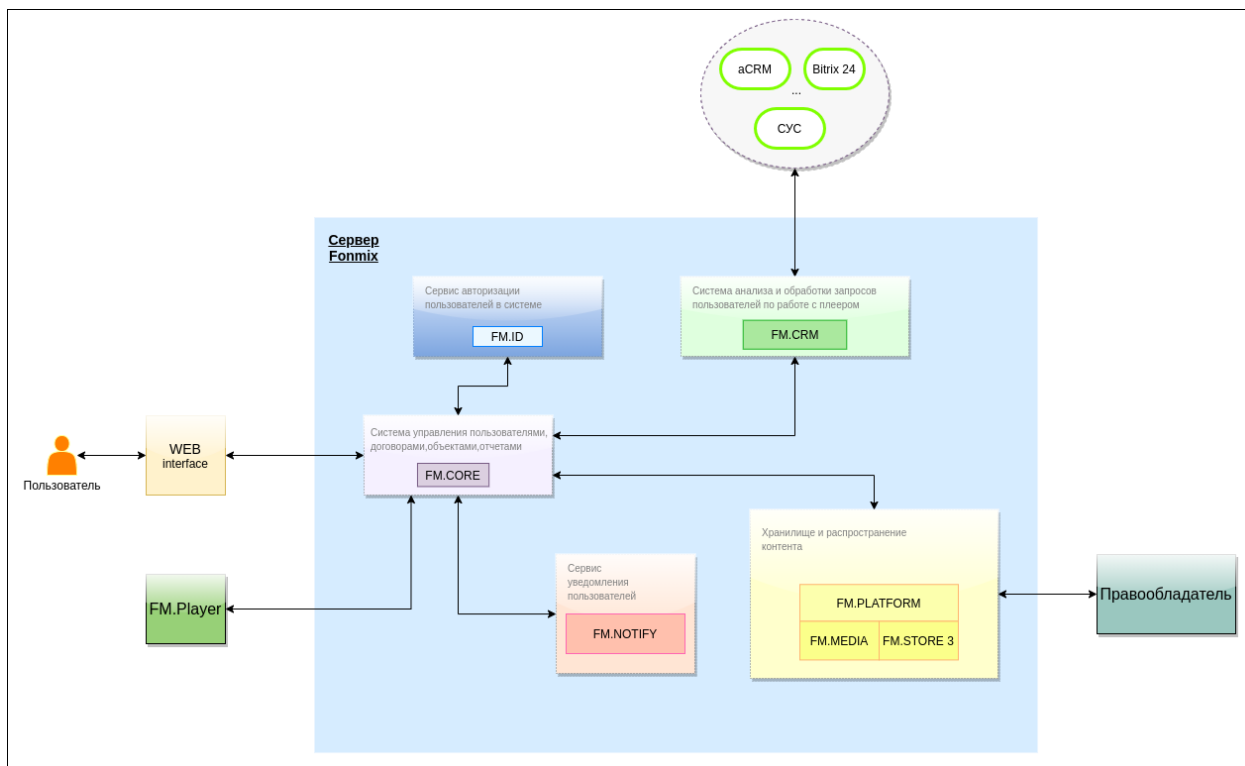


Рисунок 1.2 – Схема взаимодействие клиентов и сервера

1.1.2. Описание модели внедрения программных продуктов Agile Scrum

Текст

1.2. Исследование существующих технологий

1.2.1. Перечень функций, подлежащих автоматизации

Текст

1.2.2. Выбор и обоснование критериев качества

Для проведения сравнительного анализа аналогов и прототипов выбраны следующие критерии:

- 1) Трудозатраты на изучение технологии
- 2) Потребность в дополнительном ПО
- 3) Настраиваемость системы
- 4) Время, затрачиваемое на сопровождение документации
- 5) Публикация документации в единую справочную систему (ЕСС) компании

Критерий «Трудозатраты на изучение технологии» определяет уровень трудозатрат для сроков обучения персонала навыками владения новой технологии.

Критерий «Потребность в дополнительном ПО» определяет объем дополнительного ПО для полного сопровождения API-документации.

Критерий «Настраиваемость системы» определяет уровень трудозатрат, требуемых на первичную и дальнейшую настройку системы.

Критерий «Время, затрачиваемое на сопровождение документации» определяет продолжительность времени необходимое на сопровождение документации.

Критерий «Публикация документации в единую справочную систему компании» возможность системы в отображении документации в единой справочной системе компании. На данный момент вся программная документация по проекту Fonmix храниться вики-системе Confluence²

1.2.3. Анализ аналогов и прототипов

Рассмотрим аналоги и прототипы с точки зрения выбранных критериев качества.

² <https://ru.wikipedia.org/wiki/confluence>

1.2.3.1. Swagger

Swagger представляет собой фреймворк состоящий из нескольких отдельных, независимых утилит

- 1) Swagger Editor – онлайн редактор API-документации. Представляет собой двухоконный текстовый редактор, слева пишется документация на специальном языке разметки YAML. Графический интерфейс Swagger Editor представлен на рисунке 1.3.
- 2) Swagger UI – веб интерфейс для отображения API-документации
- 3) Swagger Codegen – автоматический генератор API-документации на основе исходного кода
- 4) Swagger Hub - предоставляет собой платное программное решение для проектирования, управления и публикации документации API.

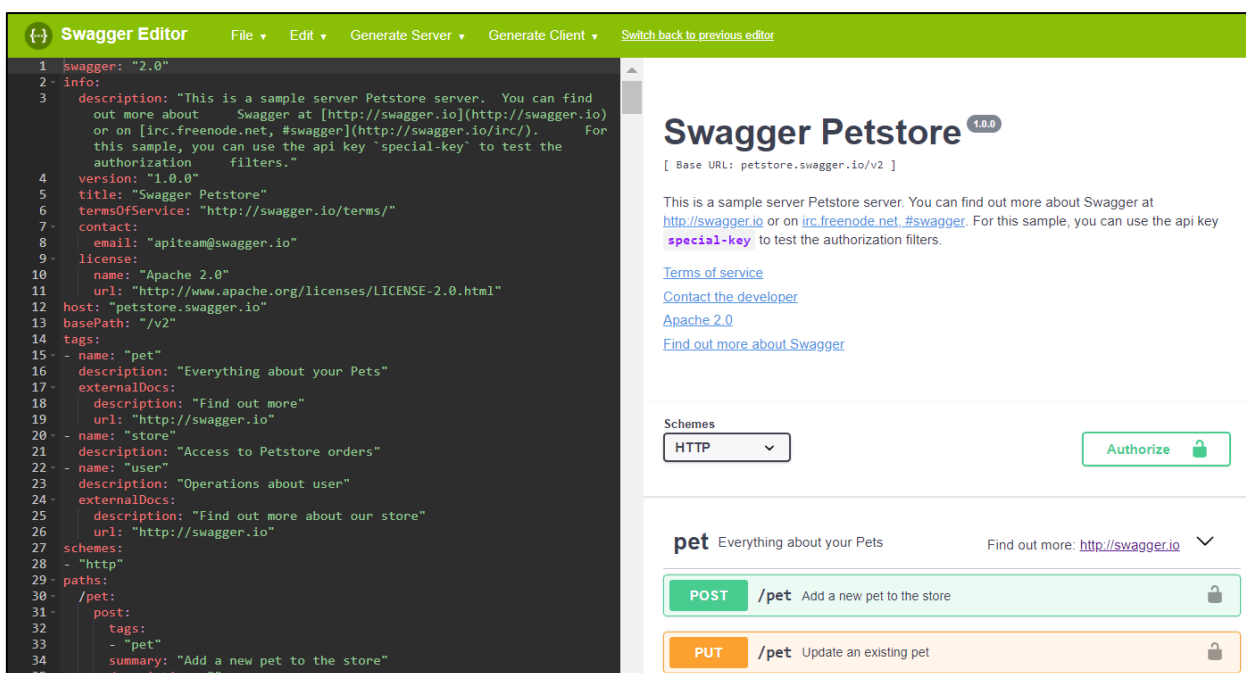


Рисунок 1.3 – Графический интерфейс Swagger Editor

Существует два подхода использования Swagger

1) Документация генерируется из комментариев в исходном коде наподобие Javadoc. Отсюда есть ряд существенных недостатков

- Код становится трудно читаем, даже если комментарии вынесены вне функций или классов
- При автоматической генерации документации необходимо настраивать CI/CD проекта

2) Написание документации отдельно от кода. Данный способ не засоряет исходный код и достаточно гибок поэтому будет рассматривать его

Перед тем как начать писать документацию, необходимо пройти учебное пособие на официальном сайте swagger.

Для того чтобы начать писать документацию необходимо открыть страницу <https://editor.swagger.io/> после чего в левой части можно будет редактировать уже готовую API-документацию.

Для написания документации на персональном компьютере, необходимо установить Swagger Editor и Swagger UI. Так как в Swagger Editor нет интерактивного взаимодействия, пользователь описывает документацию на специальном языке разметки YAML, то стоит также установить Swagger Hub.

Достоинства:

- Основным достоинством является выполнение запросов на сервер непосредственно из браузера. Swagger UI позволяет выполнить запрос и вывести ответ от сервера чтобы продемонстрировать работу API
- Автоматическая генерация клиента на разных языках программирования.
- Создания mock сервера. Это очень удобная возможность описать то как будет работать API до ее фактического написания.

Недостатки:

- Высокий порог вхождения. Необходимо изучать спецификацию Open API на которой базируется Swagger. Необходимо изучить синтаксис по работе со спецификацией Open API.
- Высока вероятность что документирование каких-то сложных API методов будет затруднительно поскольку Swagger рассчитан на базовые, простые API методы
- Явная нехватка формы обратной связи или комментариев к API методам. Если клиент захочет уточнить по поводу API метода, обратить внимание на неточность, опечатку и т.п. то скорее всего нужно будет обращаться непосредственно к разработчику API. Комментарии к документации доступны только при платной подписки на Swagger Hub

1.2.3.2. API Blueprint

API Blueprint представляет собой инструмент для ведения API-документации с использованием специального языка разметки Markdown. Отличительной особенностью от других инструментов является то что можно описывать документацию в довольно гибком формате. Из основных минусов является то что нету автоматической поддержки публикаций документации.

Графический интерфейс API Blueprint представлен на рисунке 1.4.

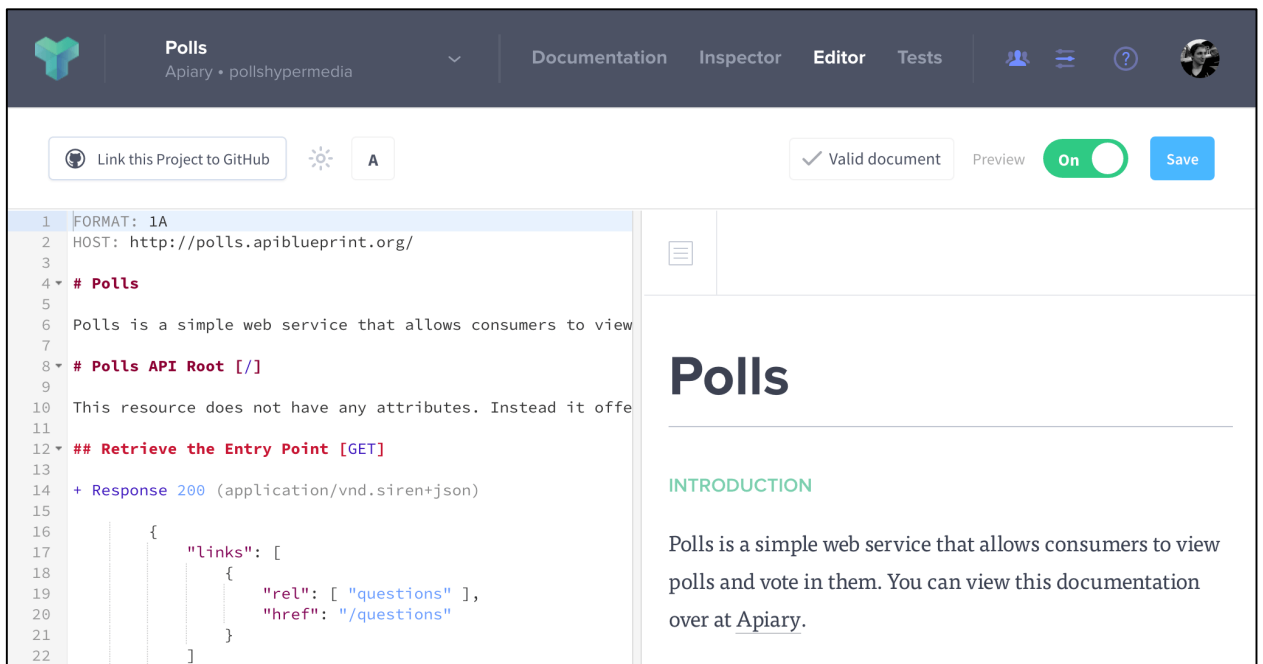


Рисунок 1.4 – Графический интерфейс API Blueprint

Достоинства:

- Удобная навигация по API-документации.
- По сравнению с Swagger у которого используется язык разметки YAML, у Blueprint используется Markdown который лучше человек читаем.
- Также, как и у Swagger есть возможность отправлять запросы на сервер из формы API-документации

Недостатки:

- Необходимо изучать язык разметки и его особенности по работе с API Blueprint
- Нету версионирования API-документации. Последняя опубликованная документация является самой актуальной и нет никакой возможности откатить ее до предыдущей версии.
- Также, как и в Swagger, нету формы обратной связи. Нету возможности связаться с автором документации чтобы уточнить детали или указать на ошибку.

1.2.3.3. RAML

RESTful API Modeling Language (RAML) – это

1.2.3.4. Ручной метод сопровождения API-документации

При ручном сопровождении документации необходимо выполнить ряд действий:

- 1) Авторизоваться в ECC Confluence
- 2) Перейти в раздел с общей технической документацией
- 3) Перейти в раздел с API-документацией проекта
- 4) Нажать на «Создать новую страницу»
- 5) Добавить необходимые компоненты на страницу
 - 5.1) Описание и название API метода
 - 5.2) Путь (URL) до API метода на сервере
 - 5.3) Описать каким образом будет осуществляться авторизация для получения доступа к API методу
 - 5.4) Описать перечень входящих параметров
 - Название параметра
 - Тип параметра
 - Указать, является данный параметр обязательным или нет
 - Указать какое значение по умолчанию установлено у параметра на сервере
 - 5.5) Примеры запросов и ответов от сервера
- 6) Выбрать «Сохранить и выйти» после чего передать готовую документацию в отдел клиентской разработки или в отдел тестирования для написания авто тестов.

Пример готовой API-документации представлено на рисунке 1.3.

Достоинства:

- Нет необходимости в приобретении дополнительного ПО
- Можно описывать документацию в любом удобном формате, однако стоит придерживаться единого формата всех документов

Недостатки:

- Время на создание и редактирование API-документации занимает очень много времени
- Необходимы дополнительные навыки по работе с Confluence

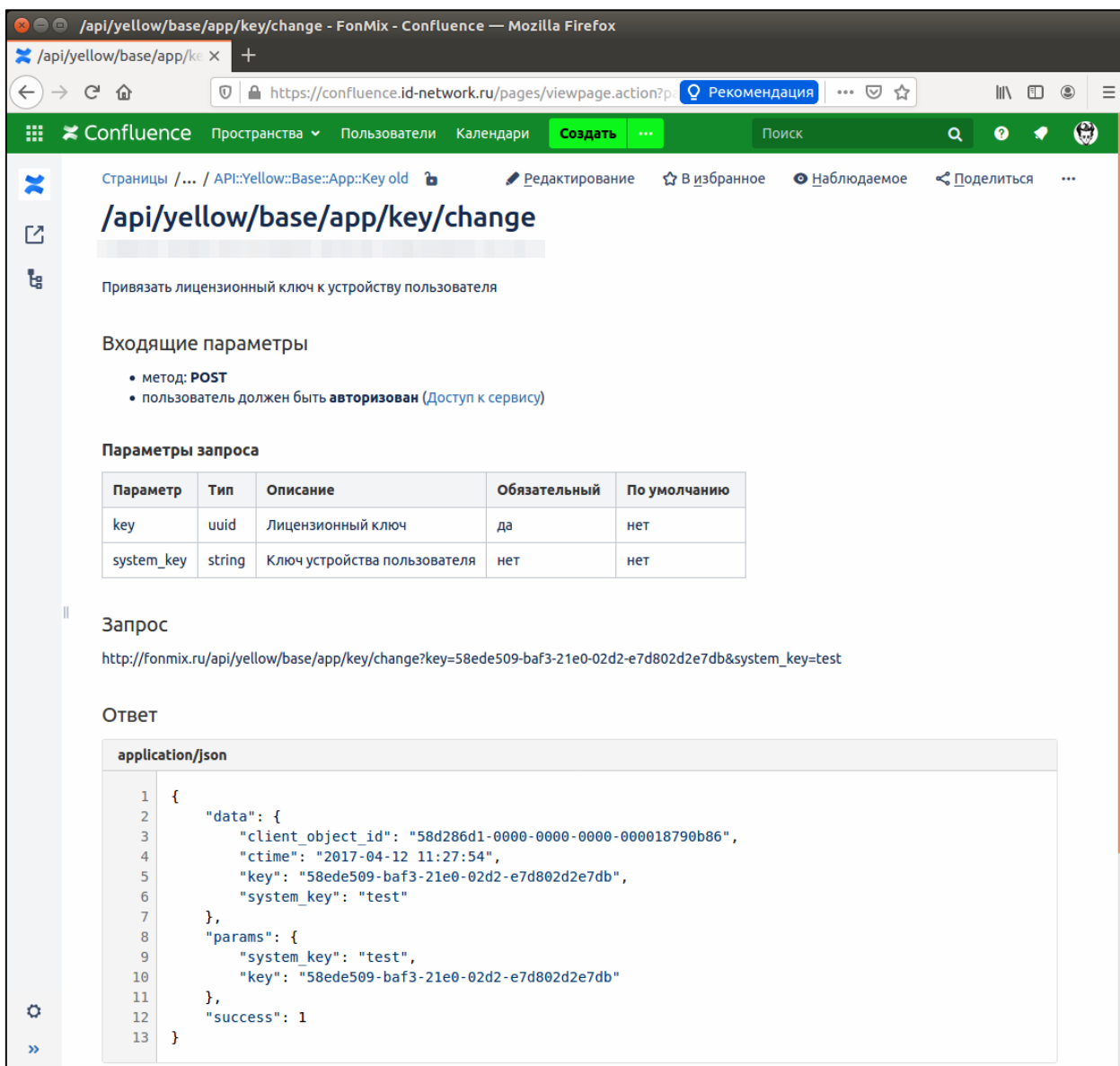


Рисунок 1.5 – Пример ручного создания API-документации

1.2.3.5. Postman

Postman представляет собой кроссплатформенное приложение с графическим интерфейсом для отправки запросов на сервер, получение ответа и его отображения.

Для установки на персональный компьютер необходимо открыть страницу в браузере <https://www.getpostman.com/>, выбрать из выпадающего

списка операционную систему (ОС), скачать и установить. Графический интерфейс Postman представлен на рисунке 1.4.

Приложение является условно бесплатным. Основной функционал доступен после авторизации на сайте.

Данный программный продукт активно используется на проекте и находится в перечне обязательных предустановленных программных продуктов компании.

Однако Postman не предоставляет возможности для документации API в единую справочную систему Confluence. Для реализации данного функционала было принято решение разработать отдельную утилиту.

Так как у Postman есть возможность экспорта и импорта всех необходимых данных можно разработать специальную утилиту, которая агрегировала бы данные и публиковала в Confluence.

Достоинства:

- Пользователю не нужно изучать дополнительные языки программирования чтобы редактировать и создавать документацию. Вся информация заполняется в интерактивных формах Postman после чего публикуется в Confluence.
- Отправка запроса на сервер и получение ответа. Демонстрация работоспособности API методов
- Экспорт и импорт коллекции для передачи сотрудникам компании
- Написание специальных скриптов для автоматического тестирования API методов.

Недостатки:

- Для реализации публикации API-документации в confluence требуется использование программного кода
- Возможны проблемы с реализацией возможных алгоритмов по сопровождению API-документации

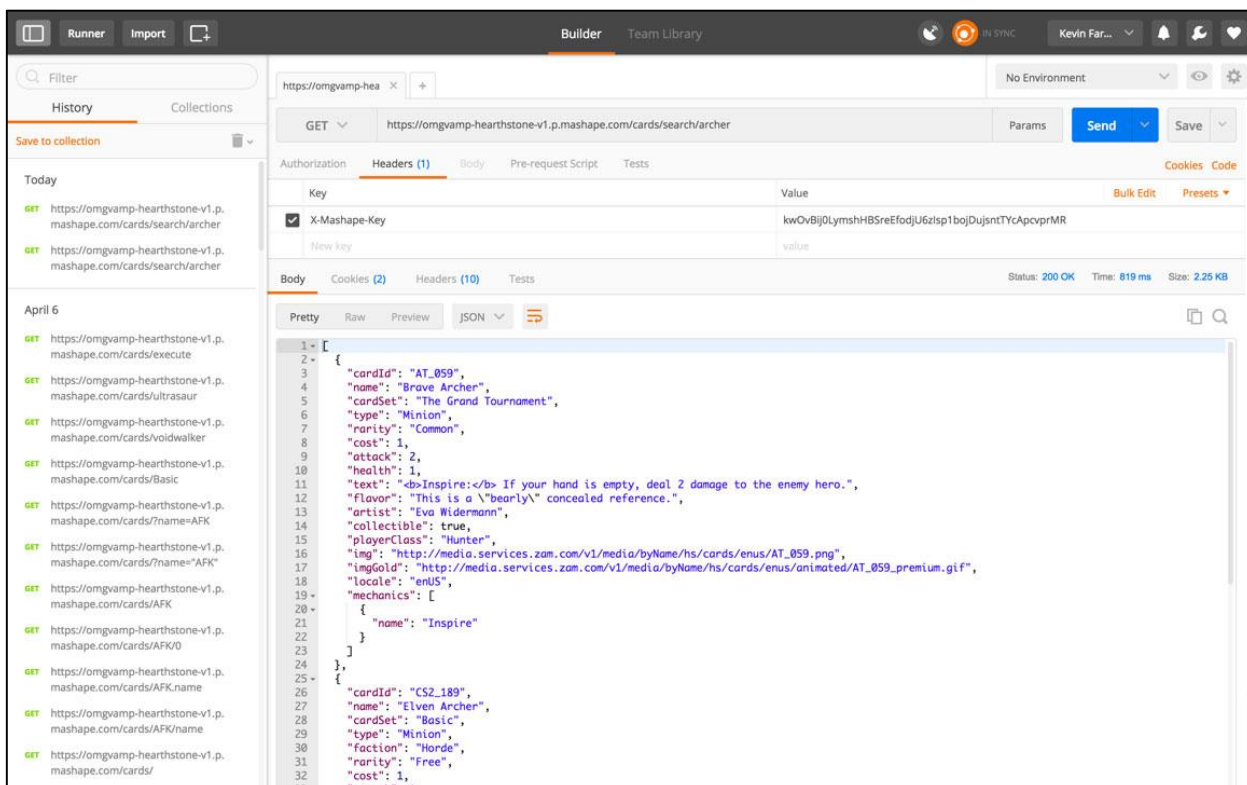


Рисунок 1.4 – Графический интерфейс Postman

1.2.4. Сравнение аналогов и прототипов

Соответствие рассматриваемых аналогов указанным критерием представлено в таблице 1.2.

В каждой ячейке стоит соответствие критерия и степень качества критерия. Степень качества и его целочисленный аналог представлен в таблице 1.2.

Таблица 1.1. – Шкала перевода степени качества критерия, в числовые

Отлично	100
Очень хорошо	80
Хорошо	60
Удовлетворительно	40

Плохо	20
Очень плохо	0

Таблица 1.2. – Качественные характеристики аналогов

	Swagger	Blueprint	RAML	Ручное сопровождение документации	Postman
Трудозатраты на изучение технологии	20	40		60	40
Потребность в дополнительном ПО	0	40		60	80
Настраиваемость системы	40	40		100	60
Время, затрачиваемое на сопровождение документации	20	40		0	80
Публикация документации в ЕСС	60	0		100	100
Σ	140	160		320	360

1.2.5. Вывод

По результатам сравнения аналогов видно, что утилита для Postman имеет наивысший балл и соответственно разработка утилиты для Postman обоснована.

2. РАСЧЕТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ

2.1. Определение требований к системе

Автоматизированная система должна обеспечивать следующих функциональных требований:

- 1) Система должна работать на персональном компьютере под управлением операционной системы Linux либо macOS
- 2) Система должна позволять заполнять всю необходимую информацию об API методах:
 - 2.1) Путь (URL) до API метода на сервере
 - 2.2) Название и подробное описание API метода
 - 2.3) Предусмотреть возможность заполнения данных об возможностях авторизации на сервере для взаимодействия с API методом.
 - 2.4) Обеспечить возможность заполнения информации о входящих параметрах API метода
 - Название параметра
 - Тип параметра
 - Указать, является данный параметр обязательным или нет
 - Указать какое значение по умолчанию установлено у параметра на сервере
 - 2.5) Система должна позволять добавлять один или несколько примеров запросов на сервер и ответов от сервера

2.6)

- 3) Должна быть удобная навигация по структуре документа или документов если их несколько
- 4) Необходимо предусмотреть поиск по документации, а также фильтрацию по: типам методов (GET, POST, PUT и т.д.), по версии документации
- 5) Необходимо обеспечить публикацию готовой документации в единой справочной системе компании

2.2. Разработка структуры автоматизированной системы

Структура взаимодействия системы выглядит следующим образом:

Пользователь, в данном случае бэкенд разработчик, в процессе разработки нового API метода проверяет его работоспособность через Postman.

По завершению кодирования функционала, заполняет необходимую информацию об API методе. Подробный список представлен в разделе «Требования к системе»

Затем добавляет примеры запросов и ответов от сервера.

В интерфейсе Postman нажимает на «Export» JSON структуры.

Выбирает место на компьютере куда происходит выгрузка структуры.

После этого запускает скрипт с указанием куда осуществлять публикацию API и путь до JSON структуры.

Авторизовывается в ECC Confluence для того чтобы проверить корректность созданной API-документации.

Передает в отдел клиентской разработки либо в отдел тестирования.

Структура и схема взаимодействия системы представлена на рисунке 2.1.

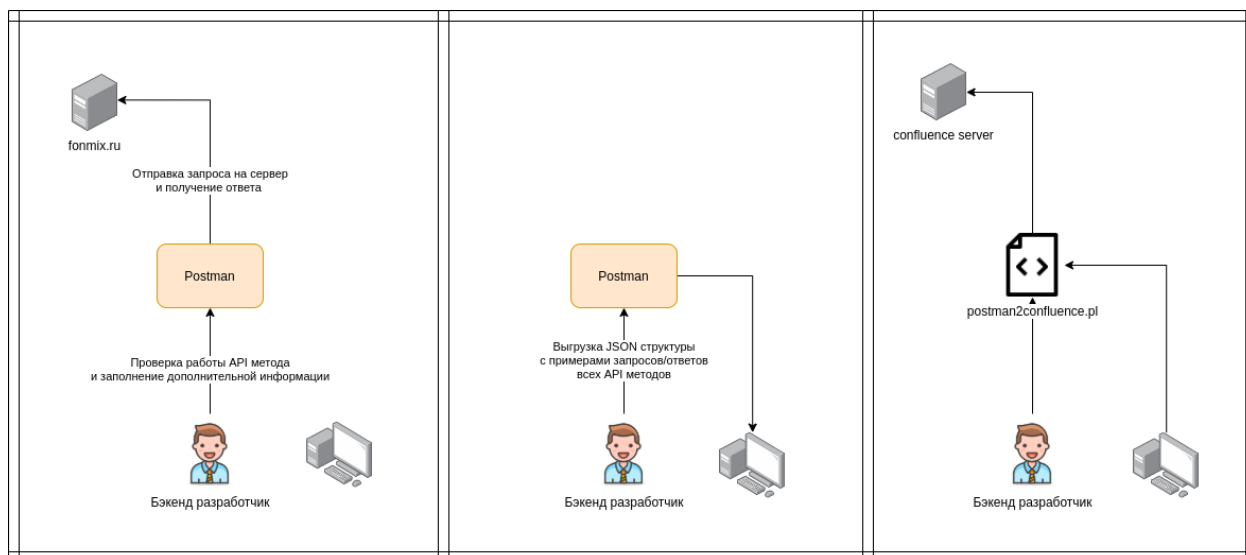


Рисунок 2.1 – Схема взаимодействия систем

2.3. Разработка структуры интерфейса взаимодействия пользователя с системой

Для взаимодействия с пользователем используется

2.4. Разработка алгоритмов программных модулей

Текст

2.5. Разработка плана проведения тестирования

Текст

3. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

3.1. Реализация разработанных алгоритмов

Текст

3.2. Тестирование и отладка системы

Текст

3.3. Руководство пользователя

Текст

ЗАКЛЮЧЕНИЕ

Что в итоге получилось.

СПИСОК ЛИТЕРАТУРЫ

1. The OpenAPI Specification – [Электронный ресурс]:
<https://github.com/OAI/OpenAPI-Specification> (Дата обращения: 24.12.2020)
2. API Blueprint – [Электронный ресурс]:
<https://apiblueprint.org/documentation> (Дата обращения: 24.12.2020)
3. API Documentation with Postman – [Электронный ресурс]:
<https://learning.postman.com/docs/publishing-your-api/documenting-your-api/> (Дата обращения: 24.12.2020)
4. RESTful API Modeling Language (RAML) – [Электронный ресурс]:
<https://raml.org/> (Дата обращения: 24.12.2020)
5. Bootstrap Documentation – [Электронный ресурс]:
<https://getbootstrap.com/docs/3.3/> (Дата обращения: 28.09.2020)
6. PostgreSQL Database Documentation – [Электронный ресурс]:
<https://www.postgresql.org/docs/> (Дата обращения: 28.09.2020)

7. Скотт Б., Нейл Т. Проектирование веб-интерфейсов. – СПб.: Символ-Плюс, 2010. – 352 с.