Тестовое задание реализовано на языке программирования Java, создана Rest API на основе картотеки супергероев. За основные параметры приняты:

1. Характеристики супергероев Marvel, которые по анализу присутствуют у всех супергероев: ID, Имя героя (name), рост (height), вес (weight), цвет глаз (eyes), цвет волос (hair), университет (universe), другие названия героя (other\_aliases), образование (education), место рождения (place\_of\_origin), ближайшие родственники (known\_relative), фото героя (img)
2. Характеристики комиксов: ID, Имя комикса (name), описание (description), дата публикации (published), фото (images), писатели (writers), художники (pencilers), музыканты (cover\_artist)

Реализовано:

1. Созданы базовые API методы:

* GET /v1/public/characters – получение персонажей с пагинацией и лексикографической сортировкой по имени
* GET /v1/public/characters/{characterId} – поиск персонажа по ID
* GET /v1/public/characters/{characterId}/comics – получение комиксов по ID персонажа с сортировкой и пагинацией
* GET /v1/public/comics - получение комиксов с пагинацией и лексикографической сортировкой по имени
* GET /v1/public/comics/{comicId} - поиск комикса по ID
* GET /v1/public/comics/{comicId}/characters - получение персонажей по ID комикса с сортировкой и пагинацией

1. Созданы POST/PUT методы для наполнения базы/файлов
2. Постраничная загрузка, сортировка, поиск
3. При создании персонажа или комикса назначается дефолтное изображение, которое в последствии загружается (заменяется) администратором.
4. Предусмотрены ответы сервера: 404 Персонаж (или комикс) не найден, 400 Bad Request,  
   200 ok , 201 created
5. Документирование API с помощью Swagger

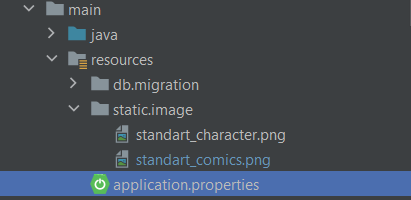
Стек технологий:

1. Язык программирования – Java 17
2. Сборка –Maven
3. Исходный код опубликован на Github
4. Framework Spring Boot
5. База данных – PostgreSQL, ORM Hibernate + Data JPA
6. Использование docker-compose для развертывания БД и запуска приложения.

Тестирование проводилось через Postman.

В рамках ТЗ мной рассматривались варианты микро-сервисной архитектуры или монолита. Программа написана в виде монолита, знаком с микро-сервисной архитектурой.

* Для запуска приложения из под IDEA указываем входные параметры : IDEA --> Run --> Edit Configuration   
  Environment variables:
* "db-password=\*\*\*\*" – пароль пользователя от БД Postgres
* "db-schema=\*\*\*\*"- схема БД
* "db-url-hosts=\*\*\*\* " – ip адрес БД
* "db-username=\*\*\*\*" – логин пользователя
* "pathToUpload= \*\*\*\* " – путь для загрузки фотографий (Копируем абсолютный путь до
* static.image в текущем проекте)
* "db-url-port=\*\*\*\*" – порт БД



**Для запуска с помощью Docker**

Заходим в файл docker-compose.yml и изменяем параметр volumes на «{Папка хранения фотографий} + : + {pathToUpload}»

ПРИМЕР : "C:\\Users\\user\\IdeaProjects\\marvel\_card\\src\\main\\resources\\static\\image\\:/marvel-app/image/"

В идеале скопировать абсолютный путь до папки static.image и подставить место {Папки хранения фотографий} .

Чтобы использовать стороннюю папку нужно будет добавить 2 фотографии из папки image сохраняя их наименование.

Подготовка к запуску:

1. Скачиваем Docker Desktop
2. Скачиваем образ postgres
3. Собираем проект с помощью сборщика maven
4. Изменяем environment по надобности.
5. Запускаем docker-compose.yml