



Факультет социально-экономических и  
компьютерных наук

Кафедра информационных  
технологий в бизнесе

Пермь,  
13.05.2025

# Разработка системы автоматической генерации тестов для задач по спортивному программированию с учётом настроек пользователя

**Автор:** Гарифуллин Александр Михайлович, студент ПИ-21-1

**Руководитель:** Ланин Вячеслав Владимирович, старший преподаватель кафедры ИТБ



Кафедра информационных технологий в бизнесе

Разработка системы автоматической генерации тестов для задач по спортивному программированию с учётом настроек пользователя

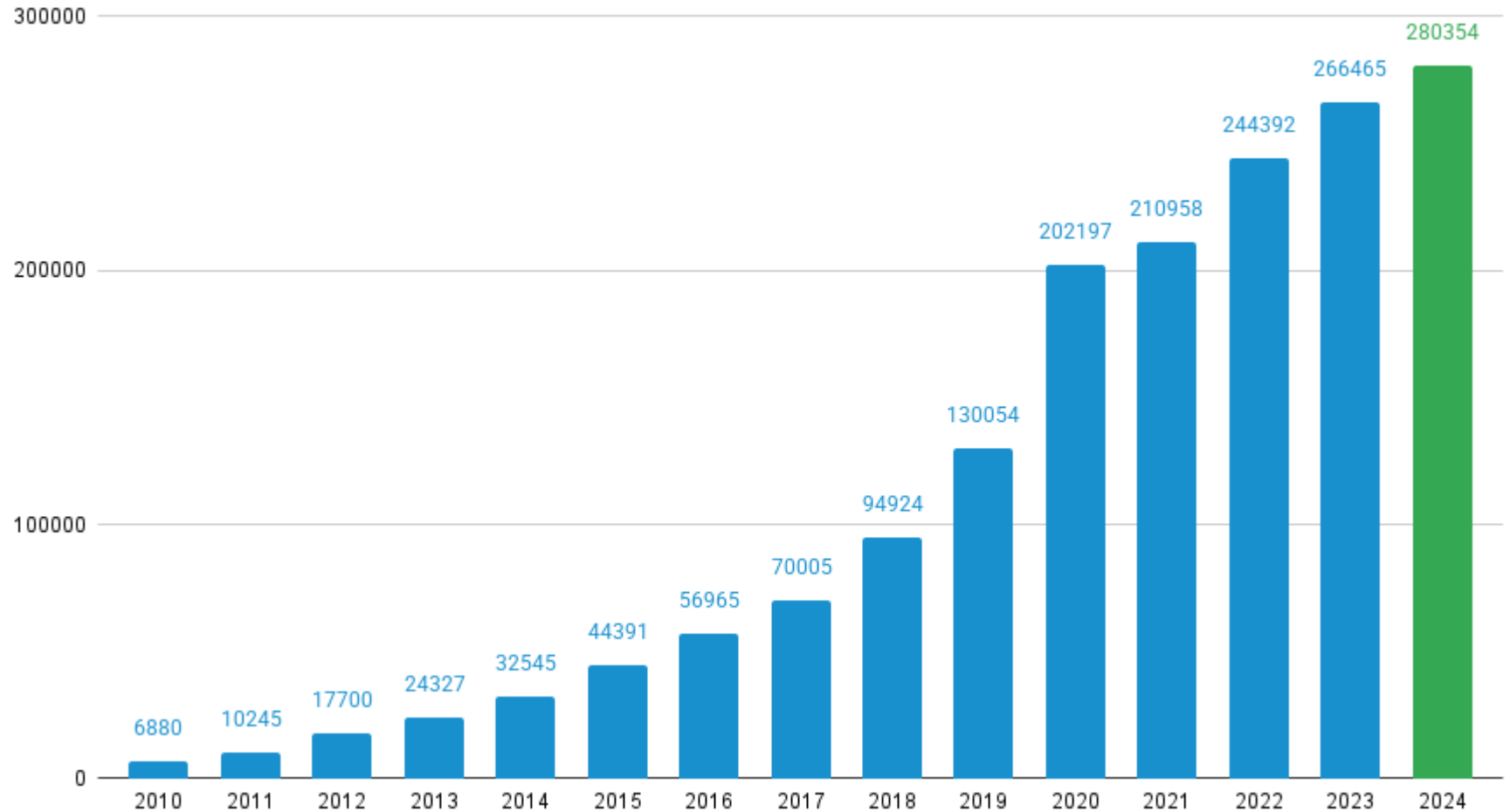
Актуальность

2/28

Актуальность



## Количество активных пользователей Codeforces



Гарифуллин Александр Михайлович



## Объект

Процесс подготовки задач по спортивному программированию

## Предмет

Программные средства, предназначенные для автоматической генерации тестов к задачам по спортивному программированию

## Цель работы

Разработка программной системы автоматической генерации тестов для задач по спортивному программированию



## Задачи

1. Анализ предметной области и формирование требований к программной системе.
2. Разработка архитектуры и проектирование программной системы.
3. Реализация программной системы и проведение её тестирования.



## Название

А. Арбуз

## Ограничения

ограничение по времени на тест: 1 second  
ограничение по памяти на тест: 64 megabytes

## Легенда

В один из жарких летних дней Петя и его друг Вася решили купить арбуз. Они выбрали самый большой и самый спелый, на их взгляд. После недолгой процедуры взвешивания весы показали  $w$  килограмм. Поспешно прибежав домой, изнемогая от жажды, ребята начали делить приобретенную ягоду, однако перед ними встала нелегкая задача. Петя и Вася являются большими поклонниками четных чисел, поэтому хотят поделить арбуз так, чтобы доля каждого весила именно четное число килограмм, при этом не обязательно, чтобы доли были равными по величине. Ребята очень сильно устали и хотят скорее приступить к трапезе, поэтому Вы должны подсказать им, удастся ли поделить арбуз, учитывая их пожелание. Разумеется, каждому должен достаться кусок положительного веса.

## Формат данных

### Входные данные

В первой и единственной строке входных данных записано целое число  $w$  ( $1 \leq w \leq 100$ ) — вес купленного ребятами арбуза.

### Выходные данные

Выведите YES, если ребята смогут поделить арбуз на две части, каждая из которых весит четное число килограмм, и NO в противном случае.

## Примеры

### Примеры

входные данные

Скопировать

8

выходные данные

Скопировать

YES

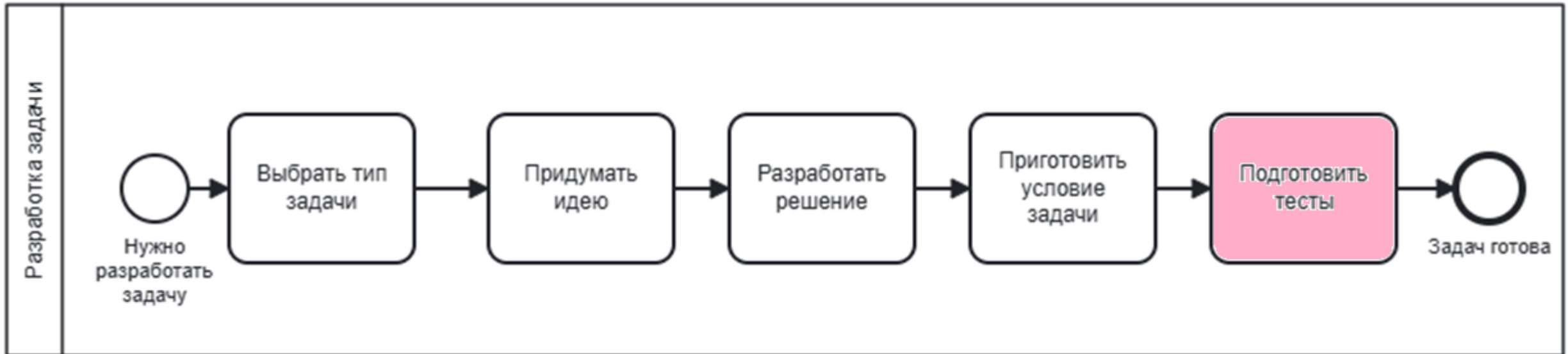
## Примечание

### Примечание

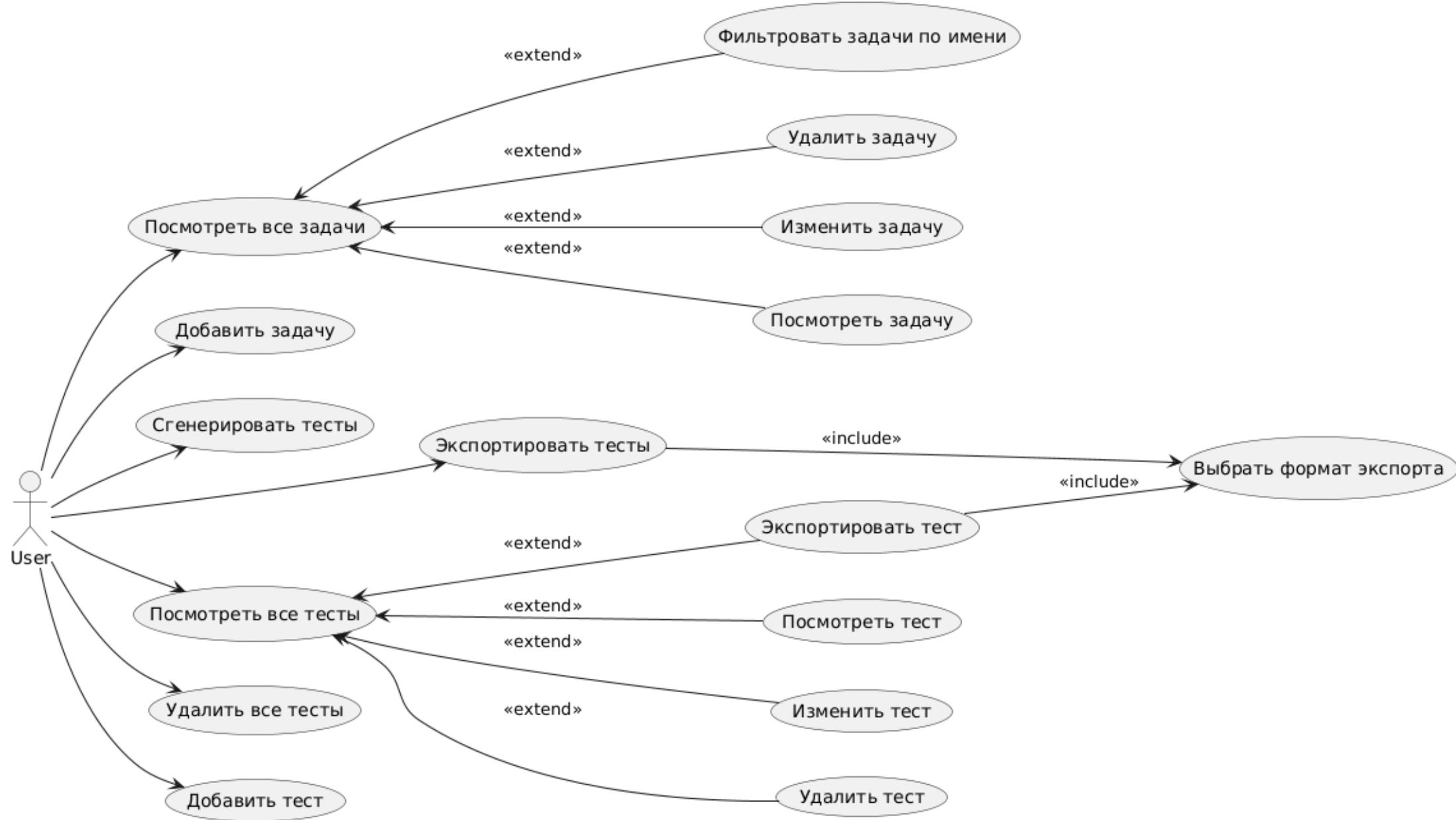
Например, ребята могут поделить арбуз на две части размерами 2 и 6 килограммов соответственно (другой вариант — две части 4 и 4 килограмма).



## AS IS



Подход	Преимущества	Недостатки
Случайная / fuzz-генерация	Быстрое покрытие, не требует знания кода	Множество не валидных тестов при сложном формате, слабое покрытие крайних случаев
Символьное выполнение	Высокое покрытие по ветвям, автоматичность	Плохо масштабируется, нужен исходный код решения
Конструктивная генерация	Гарантированная валидность, контроль автора	Спецификацию пишет человек, трудно масштабировать на много задач
Генетические алгоритмы	Находит «тяжелые» тесты, мало ложных тестов	Высокая вычислительная стоимость, неопределённое время генерации
Машинное обучение	Минимум ручных настроек, обучение на данных	Требует больших датасетов; нет строгих гарантий корректности







## Нефункциональные требования

### Безопасность:

1. Защищенные каналы обмена информации.
2. Вход через логин и пароль.
3. Хешированный пароль.

### Масштабируемость:

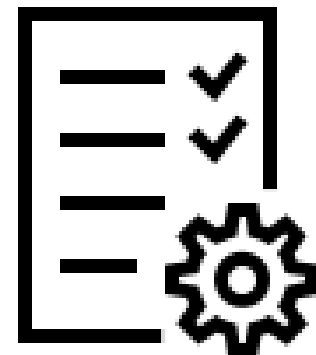
1. Динамическое распределение ресурсов.

### Модификации:

1. Поддержка добавление новых модулей для генерации тестов.

### Сопровождаемость:

1. Документированный исходный код.
2. Логирование работы системы.





## Архитектура системы

### Компоненты:

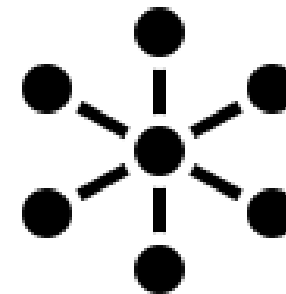
1. API-Gateway – единая точка входа.
2. Service Registry – реестр микросервисов.
3. Web-Client – веб-приложение (основной UI).
4. Export-service – микросервис для экспорта тестов.
5. Generator-service – микросервис для генерации тестов.
6. Parser-service – микросервис для обработки задач.
7. Generator-Lib – библиотека для генерации тестов.
8. Parser-Lib – библиотека для обработки задач.
9. Logging-Service – сбор, хранение и визуализация логов.

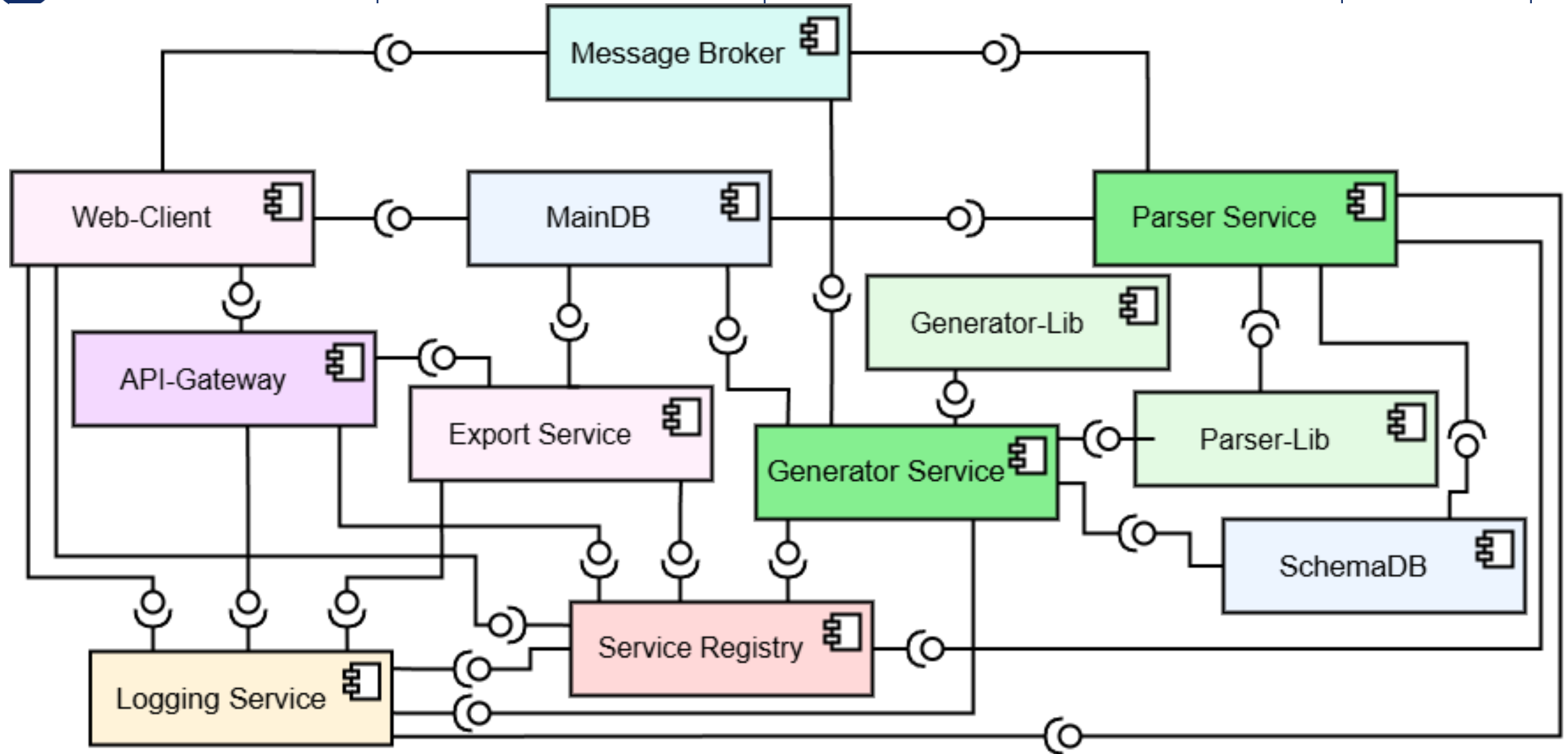
### Хранилища данных:

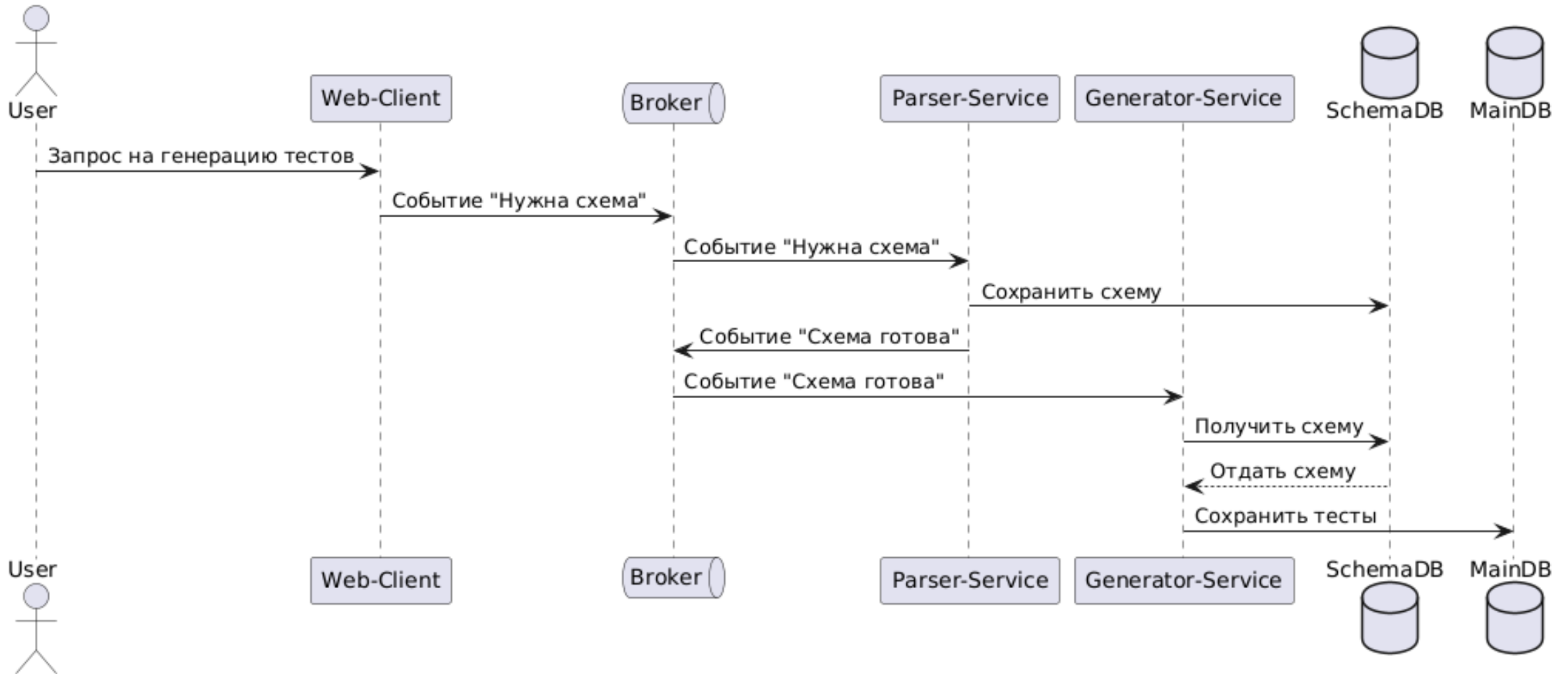
1. Реляционная СУБД – для структурированных данных
2. Документно-ориентированная хранилище – для обработанных задач.

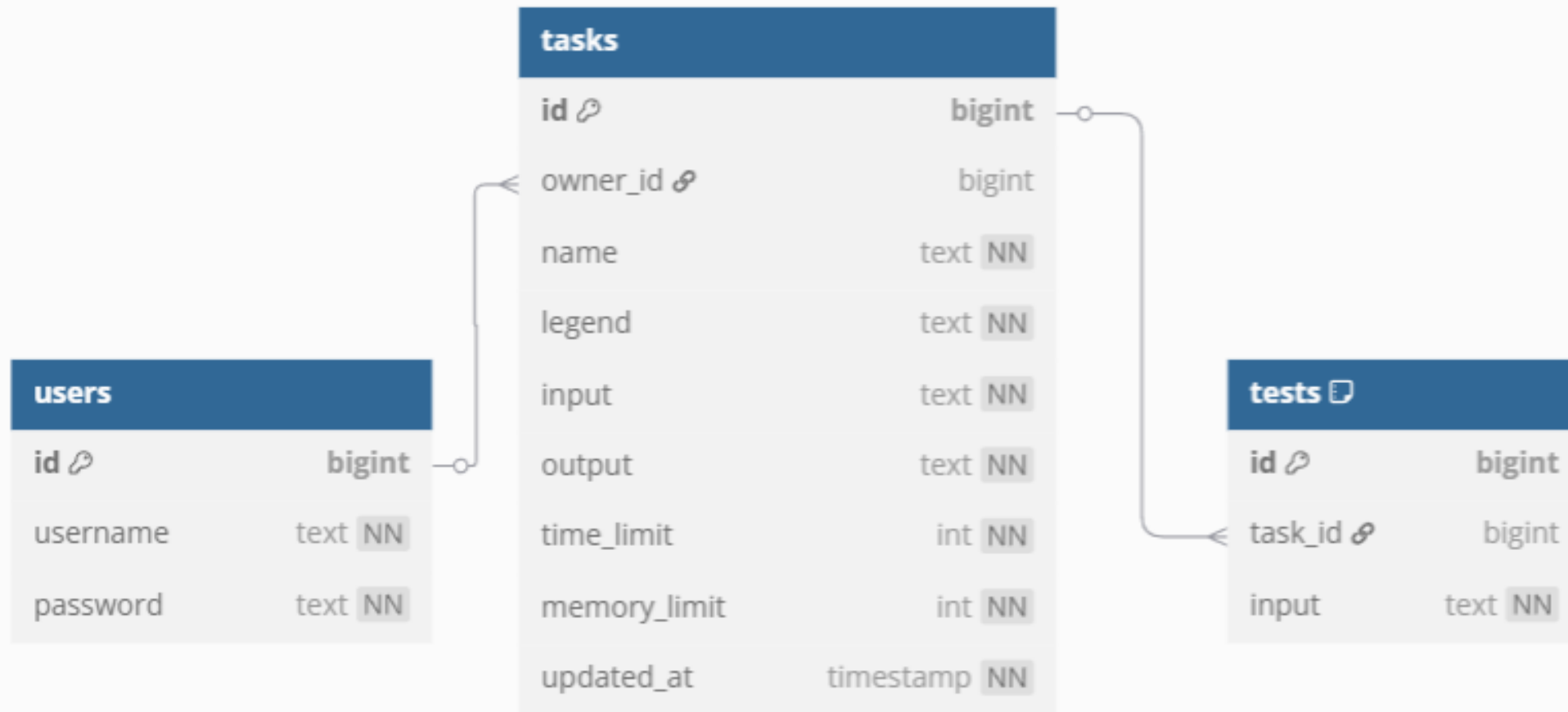
### Способы взаимодействия:

1. Синхронный – HTTP протоколы.
2. Асинхронный – брокер сообщений.





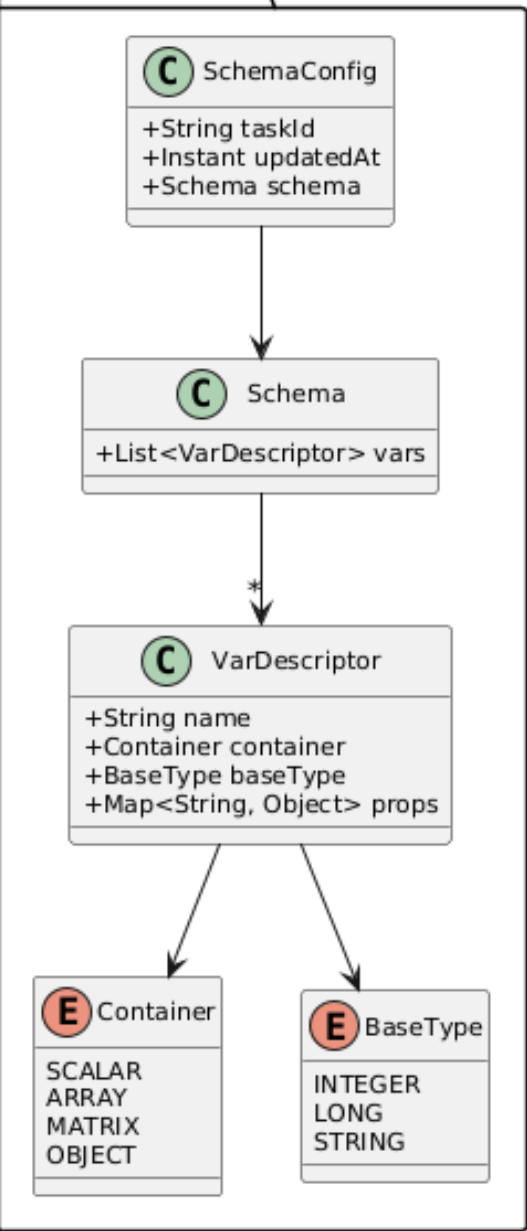






# Проектирование схемы в документо-ориентированной базе данных

## Схема задачи



Значение	Описание
SCALAR	Простое значение (одно число или строка).
ARRAY	Одномерный массив.
MATRIX	Двумерный массив (матрица).
OBJECT	Составной объект, состоящий из вложенных переменных.

Значение	Описание
INTEGER	Целое число (32-битное).
LONG	Длинное целое число (64-битное).
STRING	Строка произвольной длины.

Значение	Описание
SINGLE	Ввод всех элементов в одной строке.
MULTI	Каждый элемент на отдельной строке.

## Алгоритм обработки задач









## Анализ и выбор технологического стека

### Язык/Стек:

1. Java + Spring Boot.
2. Python + FastApi/Flask.
3. Node.js + Express.js/NestJS.
4. Go.

### СУБД:

1. PostgreSQL.
2. MongoDB.
3. MySQL.
4. Couchbase.

### Брокеры сообщений:











1. Apache Kafka.
2. RabbitMQ.
3. Redis Streams.

### Системы логирования:

1. ELK Stack (Elasticsearch + Logstash + Kibana).
2. Grafana + Loki.
3. Graylog.






## Реализация Web-Client


Технология	Назначение в системе
 Java	Основной язык реализации логики микросервиса Web-Client
 Spring Boot	Построение серверной части; реализация архитектуры MVC
 Thymeleaf	Генерация HTML-интерфейсов с динамическими данными
 Spring Security	Аутентификация и авторизация пользователей
 JWT	Формирование и хранение токена доступа в Cookie (HTTP-only)
 BCrypt	Безопасное хеширование паролей пользователей
 Spring Data JPA	Работа с реляционной БД PostgreSQL
 Liquibase	Управление версионностью схемы базы данных
 kafka	Асинхронная отправка сообщений при генерации тестов
 Spring Cloud	Централизованная маршрутизация и проксирование запросов к внутренним сервисам
 PostgreSQL	Хранение задач и тестов, созданных пользователями





# Реализация Parser-Service

  
 Spring Boot

 kafka

 Spring Data JPA

  
 MongoDB






Parser-Lib

Jackson

Технология	Назначение в системе
Java	Основной язык реализации логики микросервиса
Spring Boot	Построение микросервисной архитектуры, конфигурация и точка входа
Apache Kafka	Получение и отправка сообщений о генерации и готовности схем
Spring Data JPA	Доступ к задачам в реляционной базе PostgreSQL
PostgreSQL	Хранение описаний задач и метаданных
MongoDB	Хранение извлечённых схем входных данных
parser-lib	Библиотека парсинга: извлечение схемы из текстового описания задачи
Jackson	Сериализация и десериализация JSON-сообщений и документов



# Реализация Generator-Service

	Технология	Назначение в системе
	Java	Основной язык реализации микросервиса
 Spring Boot	Spring Boot	Базовая платформа сервиса; настройка компонентов, точка входа
 kafka	Apache Kafka	Получение уведомлений о готовности схем
Jackson	Jackson	Сериализация и десериализация Kafka-сообщений
 Spring Data JPA	Spring Data JPA	Доступ к PostgreSQL для извлечения задач и сохранения тестов
	PostgreSQL	Хранение задач и сгенерированных тестов
 MongoDB	MongoDB	Получение схем задач
Generator-Lib	generator-lib	Библиотека генерации тестов на основе схемы задачи



# Реализация Export-Service

	Технология	Назначение в системе
	Java	Язык реализации микросервиса
 Spring Boot	Spring Boot	Создание REST-контроллеров, обработка HTTP-запросов, настройка приложения
 Spring Cloud	API Gateway	Централизованная маршрутизация запросов к сервису
	PostgreSQL	Хранение задач и сгенерированных тестов, доступ к которым осуществляется при экспорте
 Spring Data JPA	Spring Data JPA	Извлечение тестов и задач из БД





# Реализация API-Gateway и Eureka-Server

Компонент	Назначение
Eureka-Server	Централизованный реестр сервисов. Отслеживает доступность и адреса микросервисов
API-Gateway	Единая точка входа. Перенаправляет внешние HTTP-запросы к нужным микросервисам



Spring Boot



Spring Cloud

Instances currently registered with Eureka			
Application	AMIs	Availability Zones	Status
API-GATEWAY	n/a (1)	(1)	UP (1)
EXPORT-SERVICE	n/a (1)	(1)	UP (1)
GENERATOR-SERVICE	n/a (1)	(1)	UP (1)
WEBCLIENT	n/a (1)	(1)	UP (1)



## Реализация Generator-Lib и Parser-Lib

### Релизованы:

1. Классы для конфигурации схемы задач.
2. Алгоритмы генерации и обработки текстового описания задачи.

### Текущая версия поддерживает:

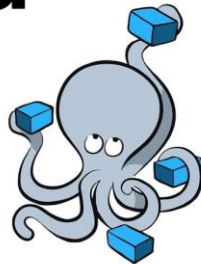
1. Мульти тесты и одиночные тесты.
2. Одна целочисленная переменная на одной строке.
3. Одномерные случайные целочисленные массивы фиксированный длины на одной строке.





## Интеграция с Kafka

Топик	Producer	Consumer	Событие
generate-tests	Web-Client	Parser-Service	Запрос генерации схемы
schema-ready	Parser-Service	Generator-Service	Уведомление о готовности схемы



docker  
Compose







## Интеграция с ELK

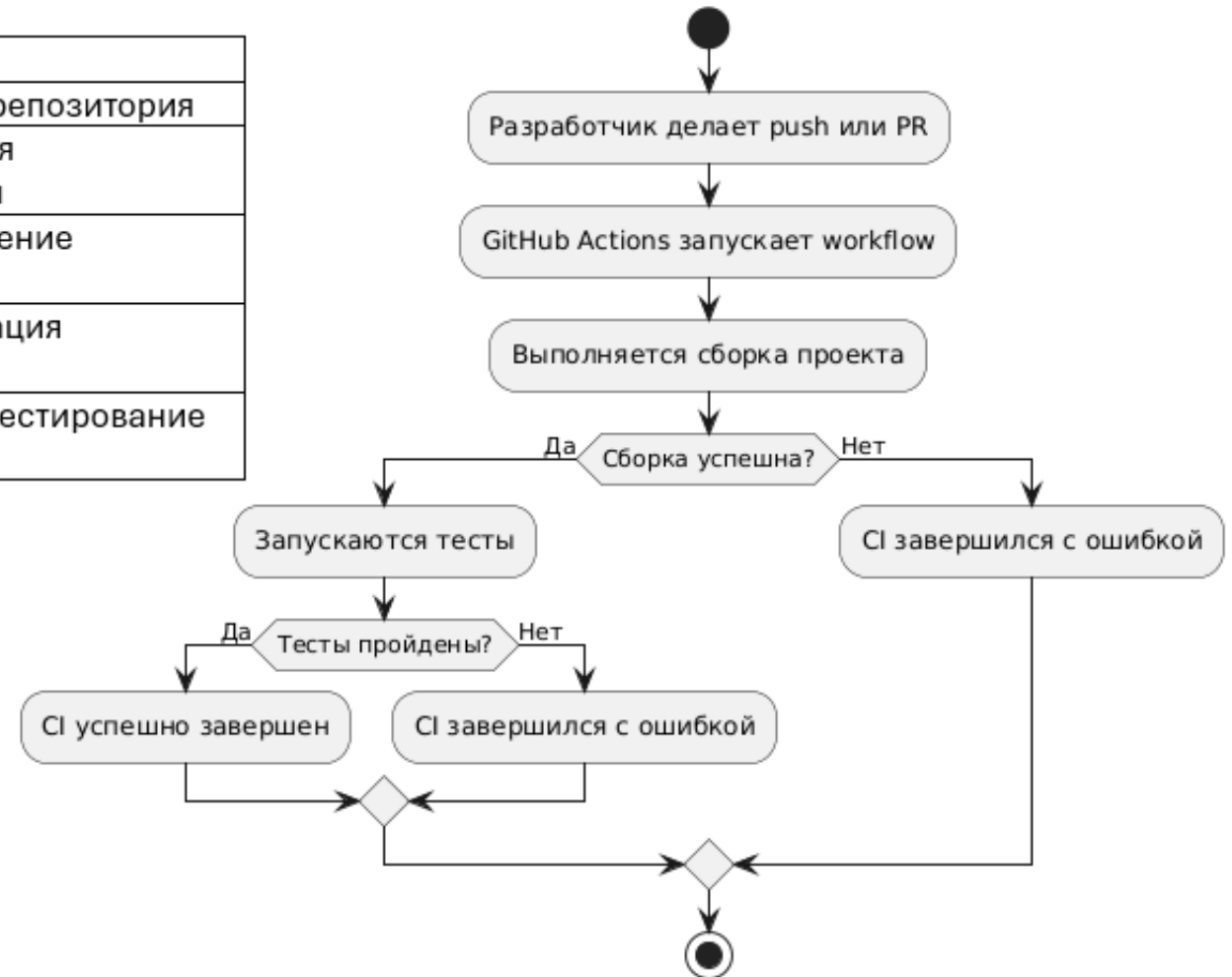


Компонент	Назначение
Elasticsearch	Хранение логов, поддержка полнотекстового поиска
Logstash	Получение логов по TCP, преобразование и отправка в Elasticsearch
Kibana	Веб-интерфейс для визуального анализа, фильтрации и поиска логов
logstash-logback-encoder	Сериализация логов в JSON для передачи в Logstash
logback-spring.xml	Конфигурация логирования микросервисов, настройка каналов вывода логов



## Организация процесса разработки и тестирования CI-процесс на GitHub Actions

Инструмент / Метод	Назначение
Git + GitHub	Контроль версий, хостинг репозитория
Именованние веток	dev/ci, dev/front, и Т.Д. для структурированной работы
Префиксы коммитов	feat, fix, ref, test - улучшение читаемости истории
Pull requests (11 шт.)	Ревью и поэтапная интеграция изменений
GitHub Actions	Автоматическая сборка и тестирование при push / PR





## Тестирование системы

Компонент	Процент покрытия кода тестами
Web-Client	49%
Parser-Service	57%
Generator-Service	31%
Export-Service	45%
Generator-Lib	81%
Parser-Lib	21%



Критерии	T1	T2	T3	T4	T5	T6	T7	T8
Классы входных данных								
Мульти тест								

Да	№	Входные данные	Ожидаемый результат	Реальный результат	+/-
Нет	1	Мульти тест $1 \leq x \leq 1e9$ $1 \leq ai \leq 1e9$	Успешная генерация	Успешная генерация	+
Ц	2	Мульти тест $1 \leq x \leq y$ $1 \leq ai \leq x$	Успешная генерация	Успешная генерация	+
Ц	3	Мульти тест $y \leq x \leq 1e9$ $x \leq ai \leq 1e9$	Успешная генерация	Успешная генерация	+
Ц	4	Мульти тест $z \leq x \leq y$ $x \leq ai \leq y$	Успешная генерация	Успешная генерация	+
Ц	5	$1 \leq x \leq 1e9$ $1 \leq ai \leq 1e9$	Успешная генерация	Успешная генерация	+
Кл	6	$1 \leq x \leq y$ $1 \leq ai \leq x$	Успешная генерация	Успешная генерация	+
Кл	7	$y \leq x \leq 1e9$ $x \leq ai \leq 1e9$	Успешная генерация	Успешная генерация	+
Кл	8	$z \leq x \leq y$ $x \leq ai \leq y$	Успешная генерация	Успешная генерация	+

Гарифуллин Александр Михайлович

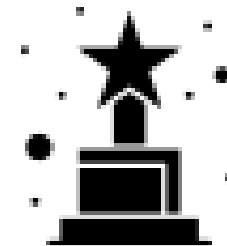


## Заключение

1. Построена модель предметной области (AS-IS).
2. Выбрана стратегия генерации тестов.
3. Сформулировано техническое задание.
4. Спроектирована микросервисная архитектура (диаграммы компонентов и последовательности).
5. Построены модели хранения: реляционная (ER-диаграмма) и документо-ориентированная (диаграмма классов).
6. Разработаны алгоритмы извлечения и генерации (диаграммы активностей).
7. Выполнен анализ технологий и выбран стек для реализации.
8. Реализовано 6 микросервисов и 2 библиотеки (Java + Spring Boot).
9. Интегрированы PostgreSQL, MongoDB, Kafka и стек ELK.
10. Настроены CI-процессы с GitHub Actions.
11. Проведено модульное и функциональное тестирование (покрытие до 81%).

## Перспективы развития

- Повышение точности генерации схем по текстовому описанию задач.
- Расширение алгоритмов генерации тестов с учетом более сложных зависимостей между входными данными.



Спасибо за внимание!  
Готов ответить на Ваши вопросы!



Контактная информация:  
amgarifullin@edu.hse.ru  
@AlexanderGarifullin  
Гарифуллин Александр  
Михайлович



### А. Триппи Троппи

ограничение по времени на тест: 1 секунда  
ограничение по памяти на тест: 256 мегабайт

Триппи Троппи живет в странном мире. Древнее название каждой страны состоит из трех слов. Первые буквы каждого слова объединяются, чтобы сформировать современное название страны.

Дано древнее название страны, пожалуйста, выведите современное название.

#### Входные данные

Первая строка содержит целое число  $t$  – количество независимых наборов входных данных ( $1 \leq t \leq 100$ ).

Следующие  $t$  строк содержат по три строки, разделенные пробелами. Каждая строка имеет длину не более 10 и содержит только строчные латинские буквы.

#### Выходные данные

Для каждого набора входных данных выведите строку, сформированную путем объединения первой буквы каждого слова.

#### Пример

входные данные	Скопировать
7	
united states america	
oh my god	
i cant lie	
binary indexed tree	
believe in yourself	
skibidi slay sigma	
god bless america	
выходные данные	Скопировать
usa	
omg	
icl	
bit	
biy	
sss	
gba	

Задача с мульти тестами

### А. Арбуз

ограничение по времени на тест: 1 second  
ограничение по памяти на тест: 64 megabytes

В один из жарких летних дней Петя и его друг Вася решили купить арбуз. Они выбрали самый большой и самый спелый, на их взгляд. После недолгой процедуры взвешивания весы показали  $w$  килограмм. Поспешно прибежав домой, изнемогая от жажды, ребята начали делить приобретенную ягоду, однако перед ними встала нелегкая задача. Петя и Вася являются большими поклонниками четных чисел, поэтому хотят поделить арбуз так, чтобы доля каждого весила именно четное число килограмм, при этом не обязательно, чтобы доли были равными по величине. Ребята очень сильно устали и хотят скорее приступить к трапезе, поэтому Вы должны подсказать им, удастся ли поделить арбуз, учитывая их пожелание. Разумеется, каждому должен достаться кусок положительного веса.

#### Входные данные

В первой и единственной строке входных данных записано целое число  $w$  ( $1 \leq w \leq 100$ ) — вес купленного ребятами арбуза.

#### Выходные данные

Выведите YES, если ребята смогут поделить арбуз на две части, каждая из которых весит четное число килограмм, и NO в противном случае.

#### Примеры

входные данные	Скопировать
8	
выходные данные	Скопировать
YES	

#### Примечание

Например, ребята могут поделить арбуз на две части размерами 2 и 6 килограммов соответственно (другой вариант — две части 4 и 4 килограмма).

Задача с одиночными тестами





Требование	Реализация в архитектуре
Безопасность	Использование защищённого HTTPS-соединения; централизованная авторизация через API-Gateway; хранение пользовательских паролей с применением криптографического хеширования в реляционной базе данных.
Масштабируемость	Асинхронная обработка задач с использованием брокера сообщений; возможность независимого горизонтального масштабирования отдельных сервисов при росте нагрузки.
Модифицируемость	Чёткое разделение логики на изолированные микросервисы; подключение новых модулей без вмешательства в уже существующие компоненты благодаря применению сервисного реестра.
Сопровождаемость	Централизованное логирование работы компонентов; наличие мониторинга и автоматической регистрации сервисов; поддержка документирования кода и трассировки выполнения.



```
{
  "@timestamp": [
    "2025-04-29T00:11:01.045Z"
  ],
  "@version": [
    "1"
  ],
  "@version.keyword": [
    "1"
  ],
  "appName": [
    "export-service"
  ],
  "appName.keyword": [
    "export-service"
  ],
  "level": [
    "INFO"
  ],
  "level_value": [
    20000
  ],
  "level.keyword": [
    "INFO"
  ],
  "logger_name": [
    "hse.diploma.service.ExportService"
  ],
  "logger_name.keyword": [
    "hse.diploma.service.ExportService"
  ],
}
```

```
{
  "message": [
    "Successfully created ZIP archive for task id 6"
  ],
  "message.keyword": [
    "Successfully created ZIP archive for task id 6"
  ],
  "service": [
    "export-service"
  ],
  "service.keyword": [
    "export-service"
  ],
  "thread_name": [
    "http-nio-auto-1-exec-6"
  ],
  "thread_name.keyword": [
    "http-nio-auto-1-exec-6"
  ],
  "_id": "45PhfpYBDVQ2oiK0-iCe",
  "_index": "microservices-logs-2025.04.29",
  "_score": null
}
```





Значение	Описание
IS_TEST_CASE_VAR	Переменная обозначает количество тестов. Тип: Boolean
MIN	Минимально допустимое значение переменной. Тип: Long
MAX	Максимально допустимое значение переменной. Тип: Long
VAR_MIN	Имя переменной, задающей минимальное значение. Тип: String
VAR_MAX	Имя переменной, задающей максимальное значение. Тип: String
ENUM_VALUES	Список допустимых значений. Тип: List (числа или строки)
GLOBAL_SUM_LIMIT	Максимальная сумма значений. Тип: Long
GLOBAL_PRODUCT_LIMIT	Максимальное произведение значений. Тип: Long
IS_UNIFORM	Все элементы одинаковы. Тип: Boolean
IS_DISTINCT	Все элементы различны. Тип: Boolean
IS_PERMUTATION	Элементы представляют перестановку. Тип: Boolean
SORTED_ORDER	Порядок сортировки. Тип: SortedOrder
MIN_LEN	Минимальная длина. Тип: Integer
MAX_LEN	Максимальная длина. Тип: Integer
VAR_MIN_LEN	Переменная, задающая минимальную длину. Тип: String
VAR_MAX_LEN	Переменная, задающая максимальную длину. Тип: String
ALLOWED_CHARS	Допустимые символы строки. Тип: String

ALLOWED_CHARS	Допустимые символы строки. Тип: String
LINE_TYPE	Способ ввода. Тип: LineType
MIN_ROW_COUNT	Мин. число строк (матрица). Тип: Integer
MAX_ROW_COUNT	Макс. число строк (матрица). Тип: Integer
VAR_ROW_COUNT	Переменная, задающая число строк. Тип: String
MIN_COLUMN_COUNT	Мин. число столбцов (матрица). Тип: Integer
MAX_COLUMN_COUNT	Макс. число столбцов (матрица). Тип: Integer
VAR_COL_COUNT	Переменная, задающая число столбцов. Тип: String
ELEMENT_VAR	Имя переменной, определяющей тип элементов. Тип: String
FIELDS	Список вложенных переменных (для объектов). Тип: List<VarDescriptor>
IS_GRAPH	Переменная представляет граф. Тип: Boolean
IS_DIRECTED	Граф ориентированный. Тип: Boolean
IS_WEIGHTED	Граф взвешенный. Тип: Boolean
IS_MULTIGRAPH	Допускаются кратные рёбра. Тип: Boolean
IS_CONNECTED	Граф связный. Тип: Boolean
IS_TREE	Граф – дерево. Тип: Boolean
ALLOW_LOOPS	Допускаются петли в графе. Тип: Boolean
RELATION	Связи между переменными. Тип: String, например $m \leq n$



## PAT\_TEST\_BLOCK:

(?ix)(?:

(?:каждый[\s\S]{0,50}?тест)|(?:первая[\s\S]{0,50}?строка))

\s\*(?:содержит|находится)\s+

(?:одно|целое\s+)?целое\s+число\s+(?<name>\w+)\s\*(

(?<range>[^\s]+)\s\*—\s\*

**RANGE\_BLOCK:** -?(?:\d+\*10^\d+|10^\d+|\d+e\d+|\d+)

**ONE\_SCALAR:** (?:одно\s+)?целое\s+число\s+(?<name>[a-zA-Z\_][a-zA-Z\_0-9]\*)

**ARR\_SCALAR:** (?<len>\w+)\s+целых\s+чисел\s+

(?<name>[A-Za-z\_])\d?

[^(\s\*(?<min>[^\s]+)\s\*<=\s\*[^\s]+\s\*<=\s\*(?<max>[^\s]+)\s\*)\s\*)

Первая строка содержит целое число  $t$  ( $1 \leq t \leq 5$ ) —  
количество тестов.

Первая строка каждого теста содержит целое  
число  $n$  ( $1 \leq n \leq 20$ ).

Вторая строка каждого теста содержит  $n$  целых  
чисел  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq n$ ).

```

1 {
2   "_id": "5",
3   "updatedAt": {
4     "$date": "2025-04-28T09:00:05.435Z"
5   },
6   "schema": {
7     "vars": [
8       {
9         "name": "t",
10        "container": "SCALAR",
11        "baseType": "INTEGER",
12        "props": {
13          "min": {
14            "$numberLong": "1"
15          },
16          "max": {
17            "$numberLong": "5"
18          },
19          "lineType": "single",
20          "isTestCase": true,
21          "fields": [
22            {
23              "name": "n",
24              "container": "SCALAR",
25              "baseType": "INTEGER",
26              "props": {
27                "min": {
28                  "$numberLong": "1"
29                },
30                "max": {
31                  "$numberLong": "20"
32                },
33                "lineType": "single"
34              },
35              "_class": "hse.diploma.model.VarDescriptor"
36            },
37            {

```

```

1   "name": "k",
2   "container": "SCALAR",
3   "baseType": "INTEGER",
4   "props": {
5     "min": {
6       "$numberLong": "1"
7     },
8     "max": {
9       "$numberLong": "7"
10    },
11    "lineType": "single"
12  },
13  "_class": "hse.diploma.model.VarDescriptor"
14},
15{
16  "name": "a",
17  "container": "ARRAY",
18  "baseType": "LONG",
19  "props": {
20    "isPermutation": false,
21    "min": {
22      "$numberLong": "1"
23    },
24    "lineType": "single",
25    "varMinLen": "n",
26    "varMax": "n",
27    "isDistinct": false,
28    "varMaxLen": "n",
29    "sortedOrder": "NONE"
30  },
31  "_class": "hse.diploma.model.VarDescriptor"
32},
33]
34}
35]
36}
37{
38  "_class": "hse.diploma.entity.SchemaConfig"
39}

```







