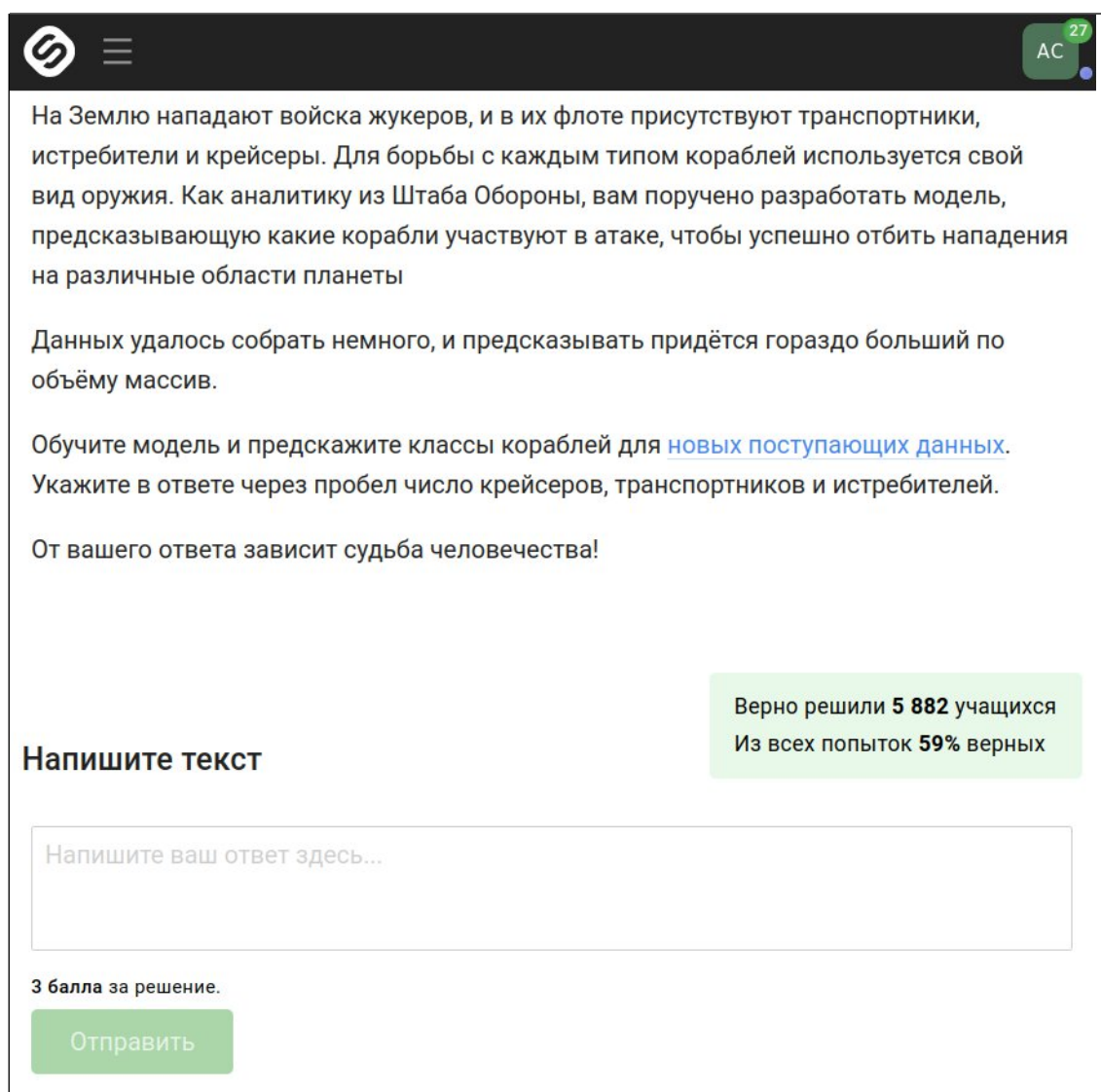


1.2.3 Проверка программ по референсным значениям

Зачастую, так как разработчики онлайн-портала не обладают достаточными ресурсами для создания подсистемы автоматизированного тестирования пользовательских программ, либо сама архитектура проверяемой программы не позволяет протестировать ее автоматически по техническим причинам (например, сама программа пользователя связана с тематикой автоматизированного тестирования, программа связана с машинным обучением и потребляет много вычислительных ресурсов и т.д.)

Пример таких заданий приведены на рисунках 4 и 5.



На Землю нападают войска жуков, и в их флоте присутствуют транспортники, истребители и крейсера. Для борьбы с каждым типом кораблей используется свой вид оружия. Как аналитику из Штаба Обороны, вам поручено разработать модель, предсказывающую какие корабли участвуют в атаке, чтобы успешно отбить нападения на различные области планеты

Данных удалось собрать немного, и предсказывать придётся гораздо больший по объёму массив.

Обучите модель и предскажите классы кораблей для [новых поступающих данных](#). Укажите в ответе через пробел число крейсеров, транспортников и истребителей.

От вашего ответа зависит судьба человечества!

Напишите текст

Верно решили **5 882** учащихся
Из всех попыток **59%** верных

Напишите ваш ответ здесь...

3 балла за решение.

Отправить

Рисунок 4 — Проверка задания на машинное обучение по референсным значениям

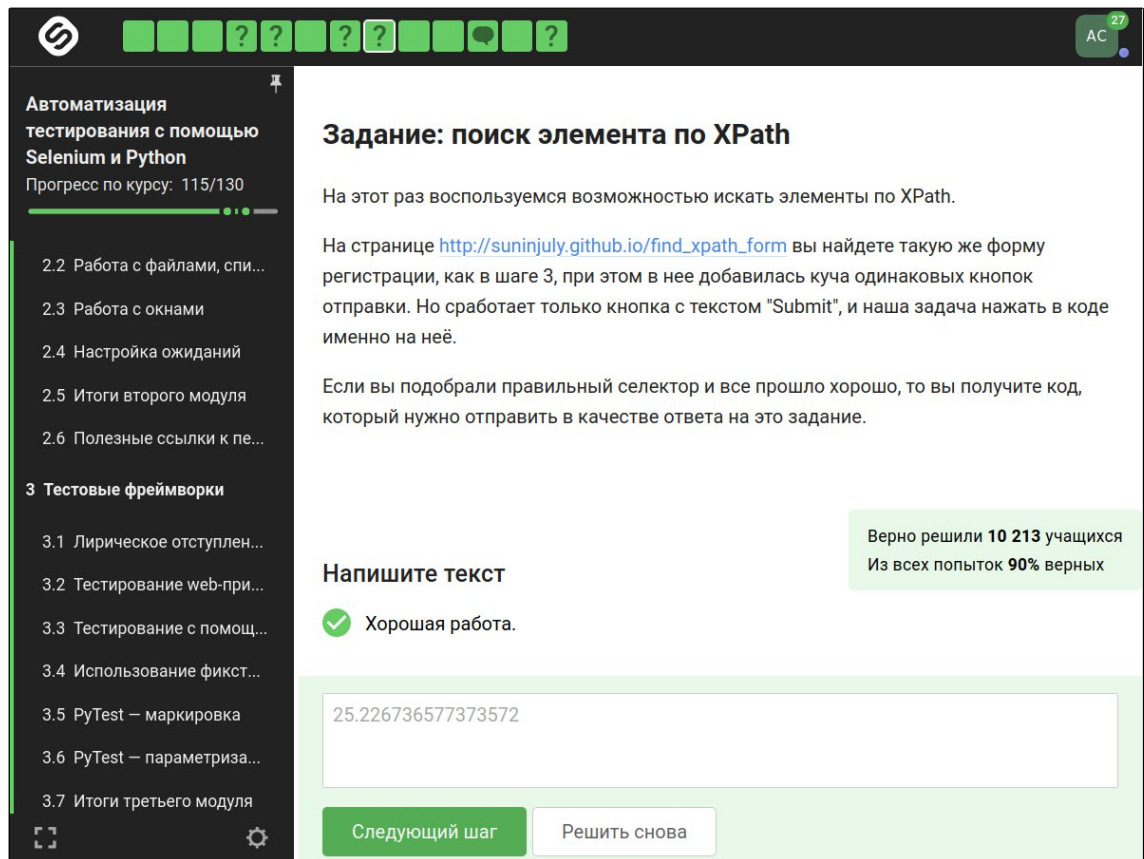


Рисунок 5 — Проверка задания на автоматизированное тестирование по референсным значениям

Функциональная модель тестирования на написание программы с проверкой по референсным значениям приведена на рисунке 6.

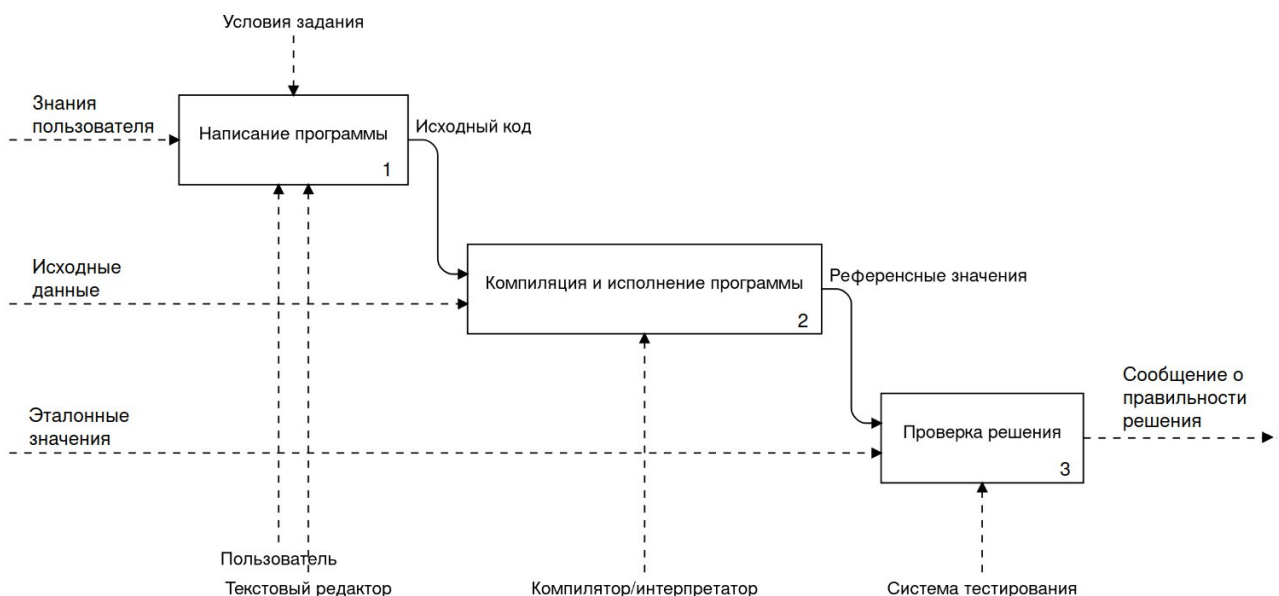


Рисунок 6 — Функциональная модель тестирования на написание программы с проверкой по референсным значениям

2 Проектирование программной подсистемы тестирования знаний языков описания аппаратуры

2.1 Проектирование архитектуры

Разработанная подсистема используется веб-приложением образовательного портала для управления учебными материалами, проверки пользовательских ответов на задания и работы со статистикой решения заданий.

Поскольку информация о пользователях используется как в разработанной подсистеме, так и в других компонентах программного обеспечения образовательного портала, БД используется совместно.

Обобщенная архитектура информационной системы показана с помощью контекст-диаграммы в нотации C4 на рисунке 9 [7].

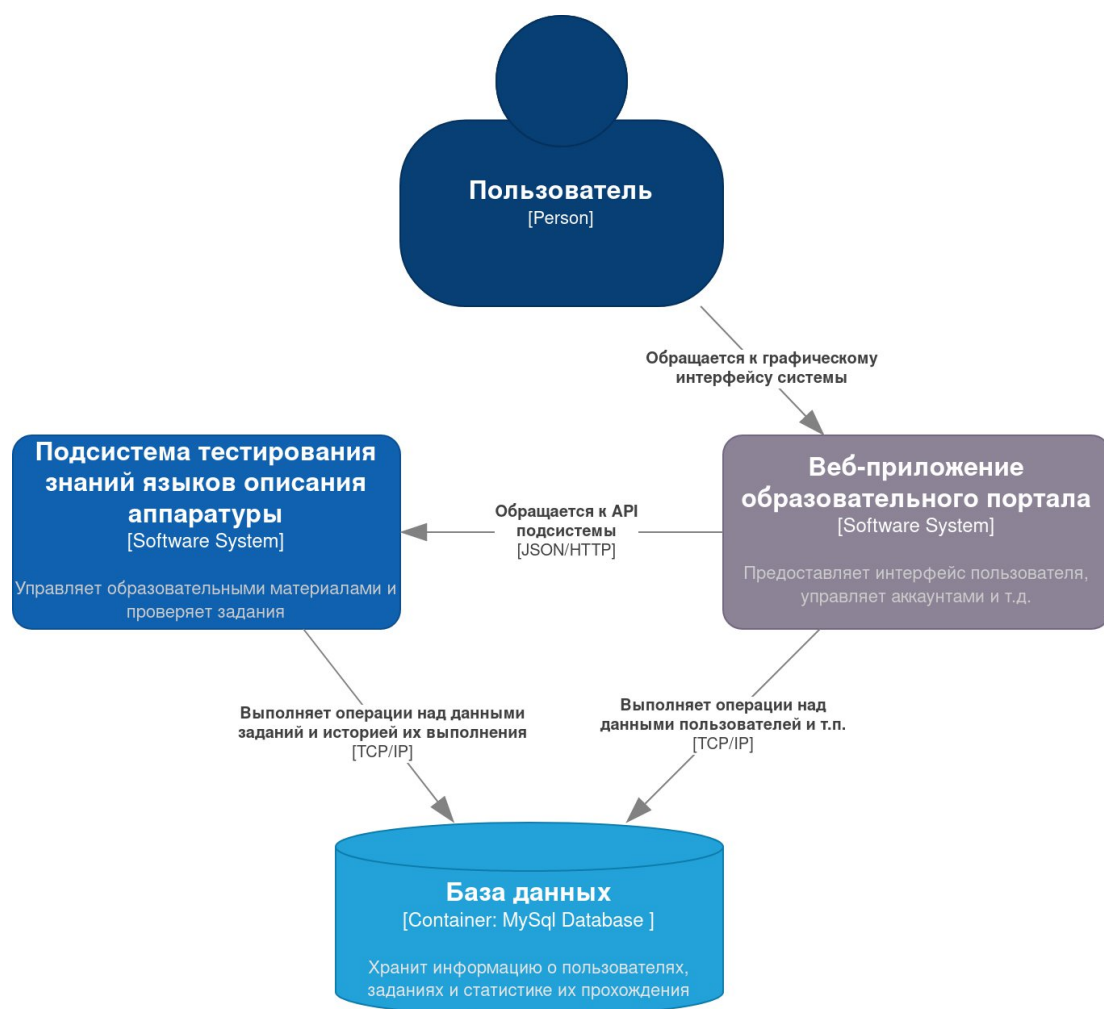


Рисунок 9 — Обобщенная архитектура информационной системы

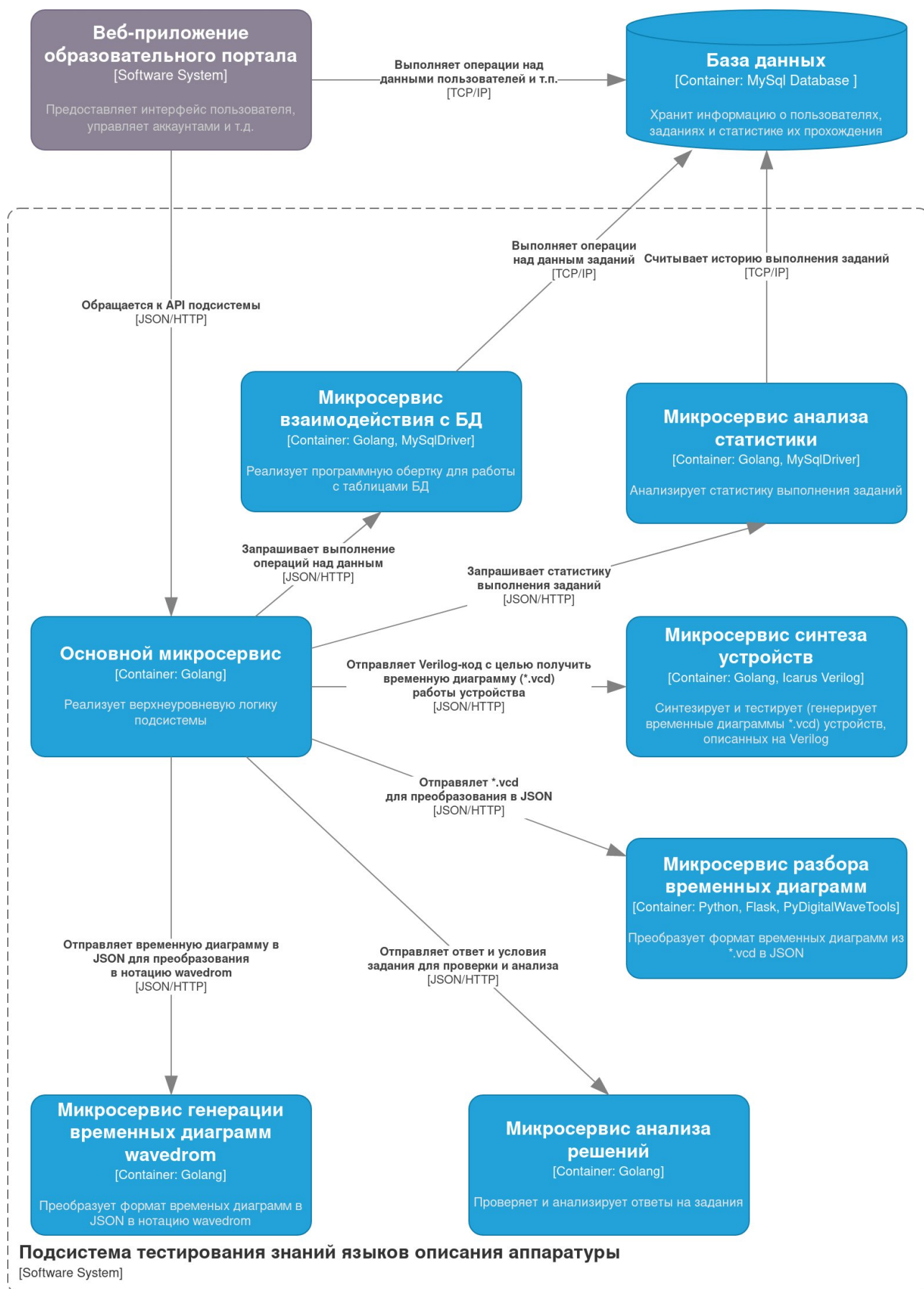


Рисунок 10 — Детализированная архитектура разработанной подсистемы

- name — имя сигнала;
- wave — форма сигнала (для каждого такта может иметь значения: «0», «1», «x», «z», «.» — сохранить предыдущее, «|» — разрыв, «=>» — обратиться к очередному элементу «data»);
- data — массив, содержащий строковые значения сигнала (можно, например, отобразить большое число для многоразрядной шины).

Пример описания временной диаграммы в формате движка Wavedrom приведен в листинге 5.

Листинг 5 — Описание временной диаграммы в формате движка Wavedrom

```
{signal: [
  {name: 'clk', wave: 'p.....|...'},
  {name: 'dat', wave: 'x.345x|=.x', data: ['0x16', '0xAA', '0x07',
'0x11']},
  {name: 'req', wave: '0.1..0|1.0'},
  {},
  {name: 'ack', wave: 'z.....|01.'}
]}
```

Визуализация данной временной диаграммы приведена на рисунке 18.

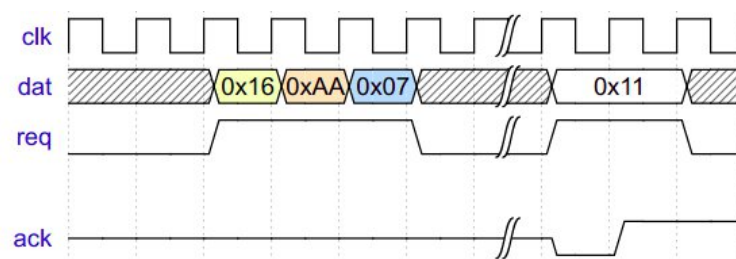


Рисунок 18 — Визуализация временной диаграммы

Основной функцией «Генератора wavedrom-диаграмм» является функция parseValues, программный код которой приведен в листинге 6.

Листинг 6 — Программный код функции parseValues

```
func (vcd_frame VCD_Struct) parseValues(end_scale int, width_scale
int) (map[string]string, map[string][]string) {
  // значения сигналов, e.g.: {"a": ["0x10", "0x35", "0xA1"], "b":
["0x03", "0x0F"]}
  var parsedData = map[string][]string{}
  // форма сигналов, e.g.: {"a": "1...0.....1...", "b":
"0....=....=."}
  var parsedWaves = map[string]string{}
  // отсортированные моменты изменения всех сигналов
```

3.4 Выбор языка программирования и библиотек для функционального тестирования

Так как написание тестов на Golang требует значительного времени и такие тесты сложнее поддерживать в силу непопулярности языка среди тестировщиков, было решено тестировать разработанные микросервисы, предварительно запустив их (см. приложение Б) и обращаясь к ним по протоколу HTTP. Такой подход позволил реализовать тесты не привязываясь к языку реализации исходного ПО.

Поскольку Python обладает простым синтаксисом, большим количеством библиотек и популярен среди тестировщиков (рисунок 21), именно он был выбран для реализации тестов [12].

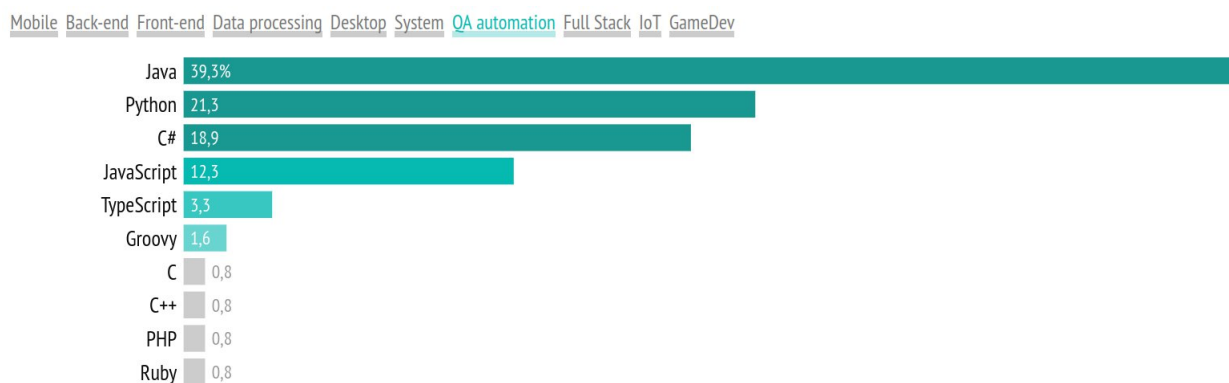


Рисунок 21 — Наиболее популярные языки в области автоматизированного тестирования

В качестве основной библиотеки для тестирования была выбрана библиотека `pytest`, являющаяся одной из наиболее популярных библиотек для автоматизированного тестирования [13].

`Pytest` обладает следующими основными преимуществами [14]:

- меньше повторяющегося кода за счет независимости от API;
- выполнение определенного набора тестов с помощью фильтрации;
- параметризация тестов — запуск одного и того же теста с разными наборами параметров;
- гибкость — архитектура библиотеки основана на плагинах, которые можно установить отдельно;

- полная обратная совместимость с unittest — возможность запуска тестов, написанных на нем;
- выполнение нескольких тестов параллельно;
- установочный код можно использовать повторно.

В дополнение к pytest была использована библиотека allure, формирующая интерактивные отчеты о прохождении тестов. Тесты в allure можно иерархически группировать и сопровождать логами и вложениями. Allure поддерживается не только для Python, но и для Java, JavaScript, Ruby, PHP, .Net и Scala.

Такой широкий набор поддерживаемых языков программирования делает allure (рисунок 22) знакомым многим разработчикам, тестировщикам и менеджерам, что упрощает поддержку тестов [15].

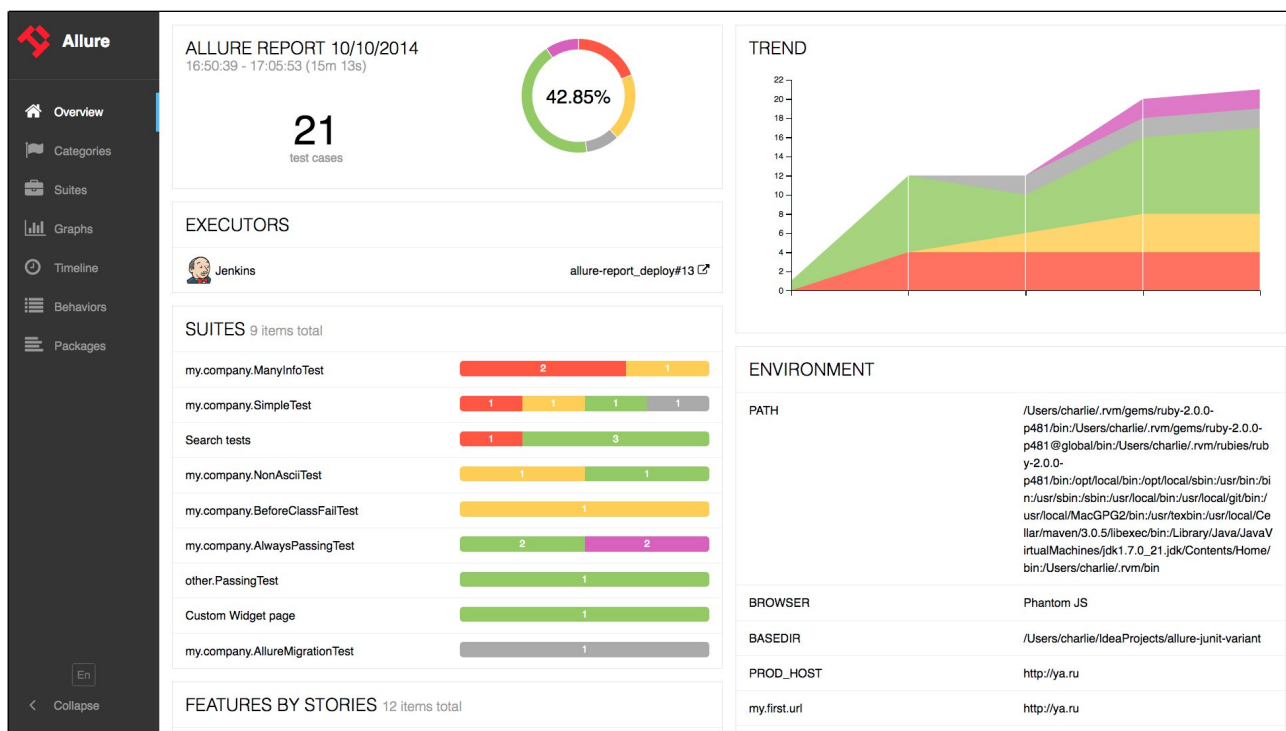


Рисунок 22 — Интерфейс allure

3.5 Реализация и проведение функциональных тестов

Для упрощения написания тестов и генерации отчетов был реализован вспомогательный модуль utils.py, отвечающий за отправку http-запросов к микросервисам, проверку http-ответов и их прикрепление к отчетам в allure. Программный код utils.py приведен в листинге 7.

Пример отчета, полученного после выполнения этого теста (и аналогичных ему) приведен на рисунке 23.

The screenshot shows the Allure Behaviors report. The left sidebar contains navigation links: Overview, Categories, Suites, Graphs, Timeline, Behaviors, and Packages. The main area displays a table of behaviors with columns for order, name, duration, and status. The status bar at the top indicates 0 failed, 0 skipped, 46 passed, and 0 not run. The table lists various tests, with '#1 test_single_correct_negative' highlighted. The right panel shows details for the selected test, including its severity, duration, description, and execution steps. The execution steps include 'Send request' and 'Check response', both of which passed.

order	name	duration	status
8	test_code_correct_negative	4ms	passed
8	test_code_correct_positive	3ms	passed
6	test_multi_correct_false_negative	3ms	passed
5	test_multi_correct_false_positive	3ms	passed
4	test_multi_correct_positive	3ms	passed
7	test_multi_error_size_mismatch	5ms	passed
1	test_single_correct_negative	5ms	passed
2	test_single_correct_positive	5ms	passed
3	test_single_error_overflow	3ms	passed

test_analyzer#test_single_correct_negative
Passed test_single_correct_negative
 Overview History Retries
 Severity: normal
 Duration: 5ms
 Description: Test for singlechoice wrong-answered task
 Execution
 > Set up
 > Test body
 > Send request 3 parameters, 3 attachments 4ms
 port '8083'
 url 'check'
 payload {'type': 'singlechoice_test', 'data': {'user_answer_id': 1, 'task': {'correct_ans...
 > Request URL 27 B
 > Request payload 639 B
 > Response payload 170 B
 > Check response [ok] 1 parameter, 1 attachment 1ms
 response <Response [200]>
 > Response payload 170 B

Рисунок 23 — Отчет о результате теста

Пример приложений к отчету приведен на рисунке 24.

The screenshot shows the Allure Behaviors report with the request and response payloads for the selected test. The left sidebar is the same as in Figure 23. The right panel displays the request URL, request payload, and response payload. The request payload is a JSON object with 'type', 'data', and 'task' fields. The response payload is a JSON object with 'status_str', 'status_code', 'message', 'is_correct', and 'data' fields.

Behaviors
 order name duration status
 Status: 0 0 46 0 0 Marks: [OK] [WARN]
 > Integrational testing 8
 > Gateway 8
 > Unit-testing 38
 > Analyzer 9
 > #9 test_code_correct_negative 4ms
 > #8 test_code_correct_positive 3ms
 > #6 test_multi_correct_false_negative 3ms
 > #5 test_multi_correct_false_positive 3ms
 > #4 test_multi_correct_positive 3ms
 > #7 test_multi_error_size_mismatch 5ms
 > #1 test_single_correct_negative 5ms
 > #2 test_single_correct_positive 5ms
 > #3 test_single_error_overflow 3ms
 > CRUD 12
 > General stats 9
 > Parser 2
 > Synthesizer 4
 > Wavedrom 2

Request URL
 http://127.0.0.1:8083/check
Request payload

```
{
  "type": "singlechoice_test",
  "data": {
    "user_answer_id": 1,
    "task": {
      "correct_answer_id": 2,
      "answers": [
        {
          "text": "Умножение",
          "hint": "Название говорит само за себя"
        },
        {
          "text": "Вычитание",
          "hint": "Перечитай главу"
        },
        {
          "text": "Сложение",
          "hint": "Все верно"
        }
      ]
    }
  }
}
```

Response payload

```
{
  "status_str": "ok",
  "status_code": 200,
  "message": "checked",
  "is_correct": false,
  "data": {
    "hint": "Перечитай главу"
  }
}
```

 > Check response [ok] 1 parameter, 1 attachment

Рисунок 24 — Приложения к отчету

После реализации аналогичным образом всех тестов из таблиц 4-9 они были запущены.

Благодаря параллельному запуску тестов через плагин xdist (в данном случае — в 2 потока) удалось выполнить все тесты менее, чем за 600 мс (рисунок 25).



Рисунок 25 — Хронология запуска тестов

Статистика запуска тестов из отчета allure приведена на рисунке 26.

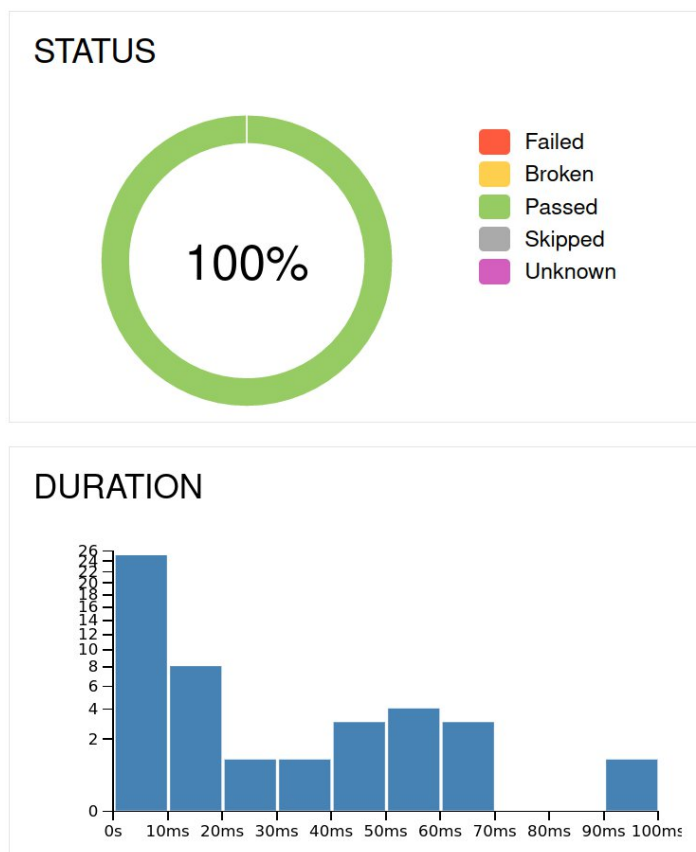


Рисунок 26 — Статистика запуска тестов

Все тесты завершились успешно, о чем свидетельствует приведенная статистика.

визуализировать статистику, допустимое среднее время ответа для запросов к БД было установлено равным 300 мс, а 95 перцентиль — 500 мс.

Результаты нагрузочного тестирования микросервиса разбора временных диаграмм приведены в таблице 10.

Таблица 10 — Результаты нагрузочного тестирования микросервиса разбора временных диаграмм

Статистика запросов							
Запросы	Ошиб.	Среднее (мс)	Мин. (мс)	Макс. (мс)	Сред. размер (байт)	RPS	Ошибки / с
18458	0	261	3	706	493	279.9	0.0
Статистика ответов							
50%ile (мс)	60%ile (мс)	70%ile (мс)	80%ile (мс)	90%ile (мс)	95%ile (мс)	99%ile (мс)	100%ile (мс)
260	300	350	400	450	470	520	710

Изменения частоты запросов, времени ответа и количества пользователей показаны на рисунках 27-28.



Рисунок 27 — Результаты нагрузочного тестирования микросервиса разбора временных диаграмм



Рисунок 28 — Результаты нагрузочного тестирования микросервиса разбора временных диаграмм

Результаты нагрузочного тестирования синтезатора приведены в таблице 11.

Таблица 11 — Результаты нагрузочного тестирования синтезатора

Статистика запросов							
Запросы	Ошиб.	Среднее (мс)	Мин. (мс)	Макс. (мс)	Сред. размер (байт)	RPS	Ошибки / с
12936	0	525	13	1339	3050	152.8	0.0
Статистика ответов							
50%ile (мс)	60%ile (мс)	70%ile (мс)	80%ile (мс)	90%ile (мс)	95%ile (мс)	99%ile (мс)	100%ile (мс)
580	610	640	680	770	870	950	1300

Изменения частоты запросов, времени ответа и количества пользователей показаны на рисунке 29.

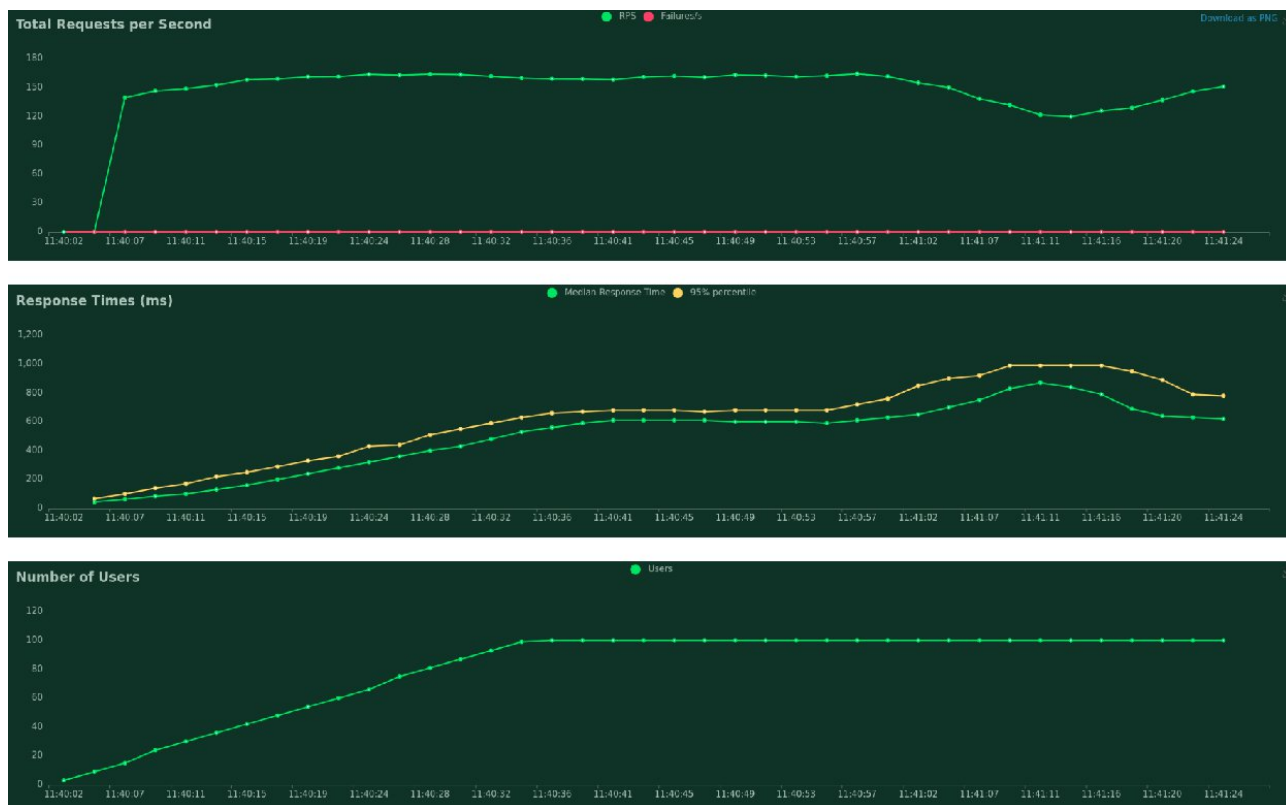


Рисунок 29 — Результаты нагрузочного тестирования синтезатора

Результаты нагрузочного тестирования обращения к БД через слой бизнес-логики приведены в таблице 12.

Таблица 12 — Результаты нагрузочного тестирования обращения к БД через слой бизнес-логики

Статистика запросов								
Марш.	Запросы	Ошибки	Среднее (мс)	Мин. (мс)	Макс. (мс)	Сред. размер (байт)	RPS	Ошибки / с
/levels	6743	0	85	3	676	506	111.8	0.0
/stats	20486	0	161	2	1151	331	339.5	0.0
Итого	27229	0	142	2	1151	374	451.3	0.0

Продолжение таблицы 12

Статистика ответов								
Марш.	50%ile (мс)	60%ile (мс)	70%ile (мс)	80%ile (мс)	90%ile (мс)	95%ile (мс)	99%ile (мс)	100%ile (мс)
/levels	78	94	110	130	160	180	220	680
/stats	110	140	190	270	370	460	630	1200
Итого	100	130	160	210	330	430	600	1200

Изменения частоты запросов, времени ответа и количества пользователей показаны на рисунке 30.

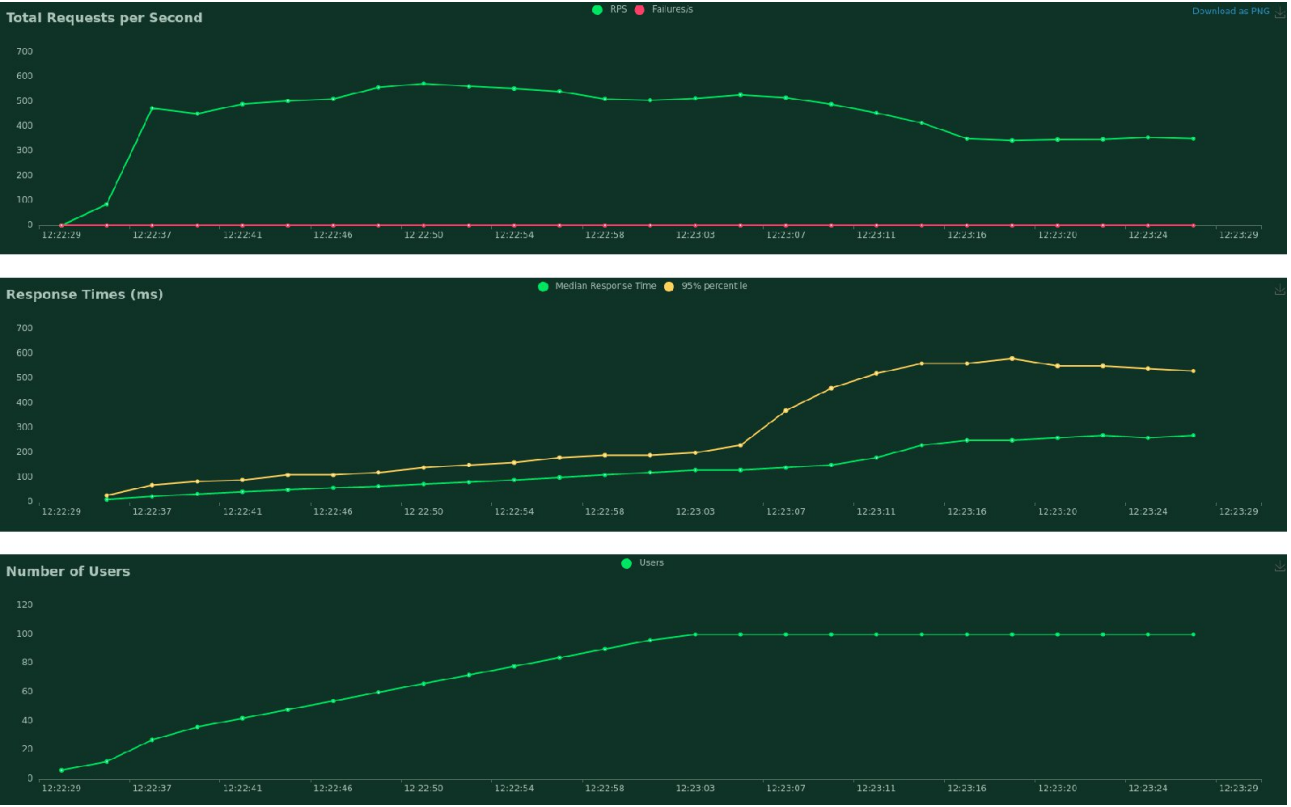


Рисунок 30 — Результаты нагрузочного тестирования обращения к БД через слой бизнес-логики

Тестируемые компоненты в ходе тестирования работали корректно, полученные задержки находятся в допустимых пределах.

2 Структура программы

Для реализации Программной подсистемы тестирования знаний языков описания аппаратуры разработано программное обеспечение в соответствии со следующей компонентной структурой.

Для реализации Программной подсистемы тестирования знаний языков описания аппаратуры применены принципы микросервисной архитектуры. Подсистема предназначена для интеграции в информационную систему образовательного портала (Рисунок 1).



Рисунок 1 — обобщенная архитектура информационной системы образовательного портала

В представленной информационной системе обращение к Программной подсистеме тестирования знаний языков описания аппаратуры происходит в стиле REST, БД используется совместно с Web-приложением образовательного портала.

В состав Подсистемы входят следующие функциональные блоки:

- 1) основной микросервис;
- 2) микросервис взаимодействия с БД;
- 3) микросервис синтеза устройств (синтезатор);
- 4) микросервис разбора временных диаграмм;
- 5) микросервис генерации временных диаграмм wavedrom;

6) микросервис анализа решений (анализатор);

7) микросервис анализа статистики.

Детализированная архитектура Подсистемы (в составе информационной системы) изображена на рисунке 2.

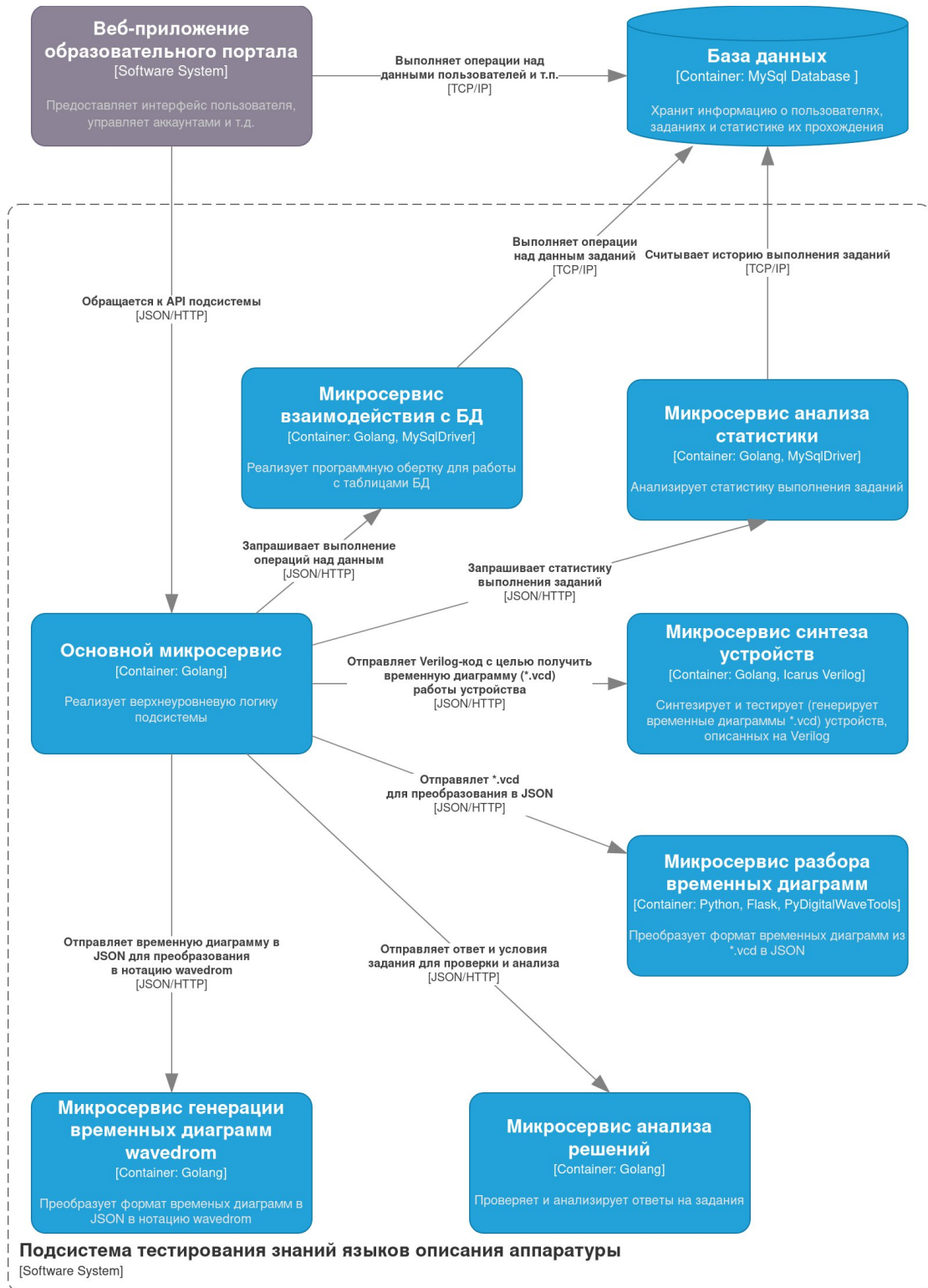


Рисунок 2 — архитектура рассматриваемой подсистемы

После чего откроется веб-страница с интерактивным отчетом о результатах тестов (Рисунок 4).

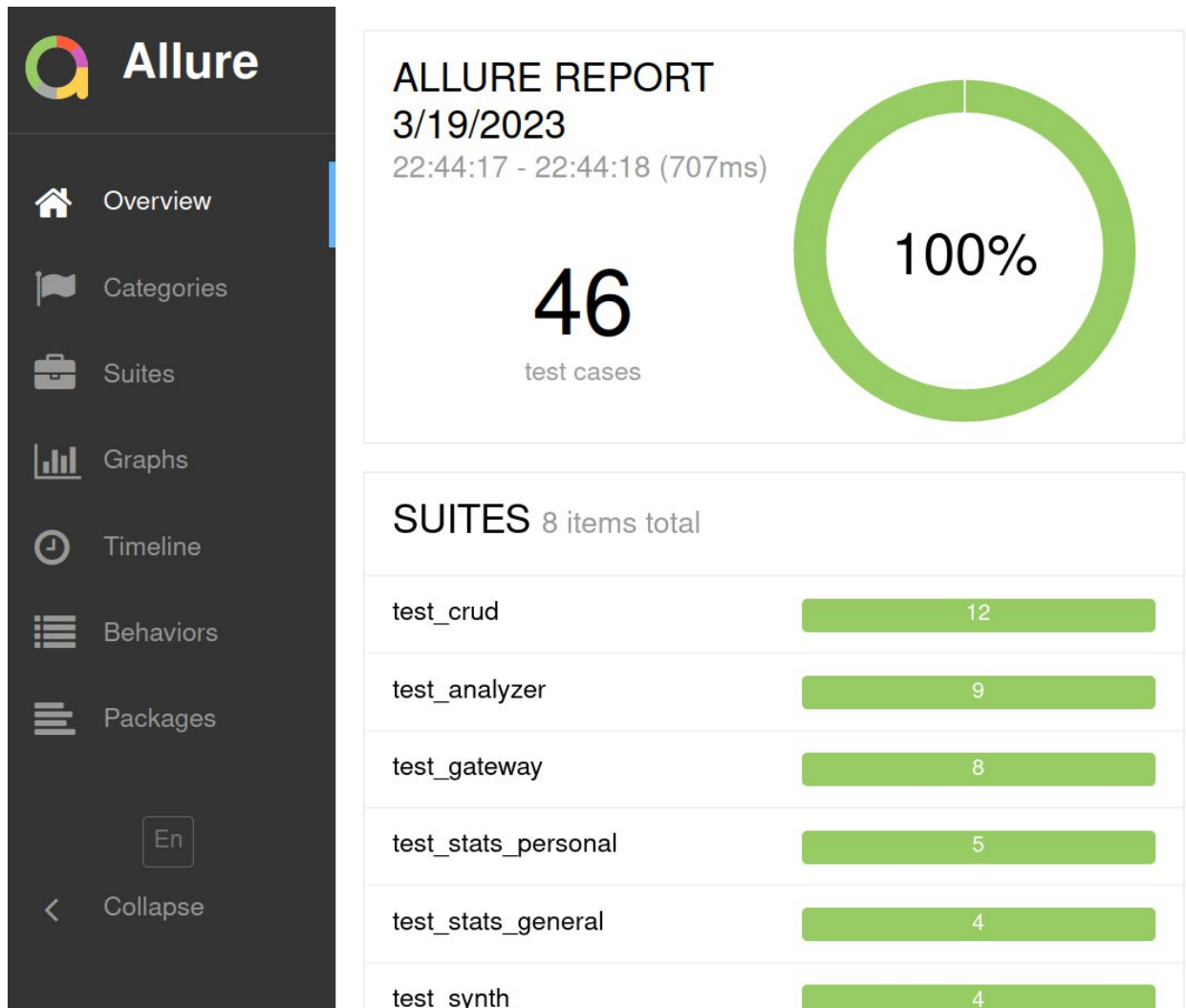


Рисунок 4 — интерактивный отчет о результатах тестов