



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

Выпускная квалификационная работа

по теме:

«Проектирование и разработка автоматизированных тестов для проекта кредитного конвейера»

Выполнил: ст. гр. ПИпб-116 Тимин С.И.

Научный руководитель: доц.каф. ВТиСУ Шутов
А.В.

Владимир, 2020

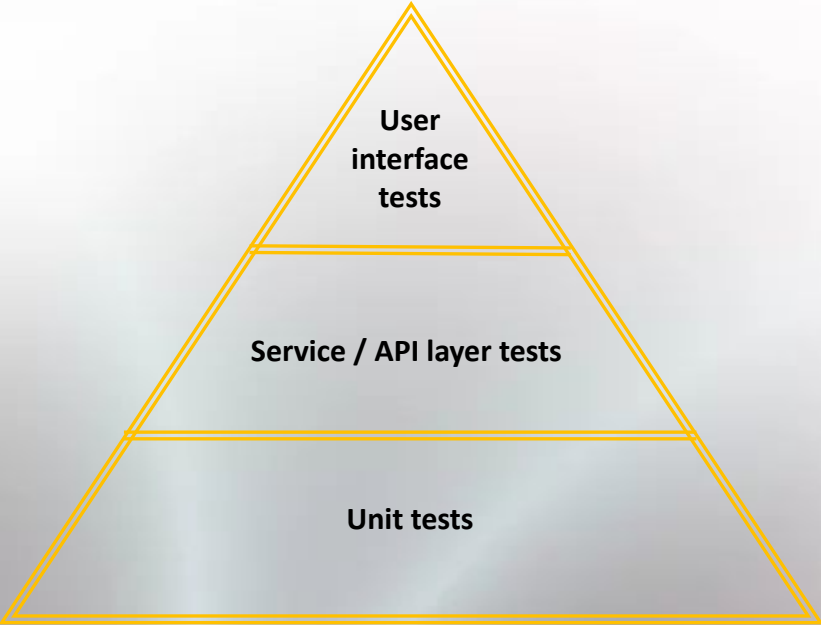
Цель и задачи ВКР

Цель выпускной квалификационной работы: повышение эффективности тестирования путем автоматизации процесса на проекте кредитного конвейера, разработка тестовых скриптов для компонентного интеграционного тестирования.

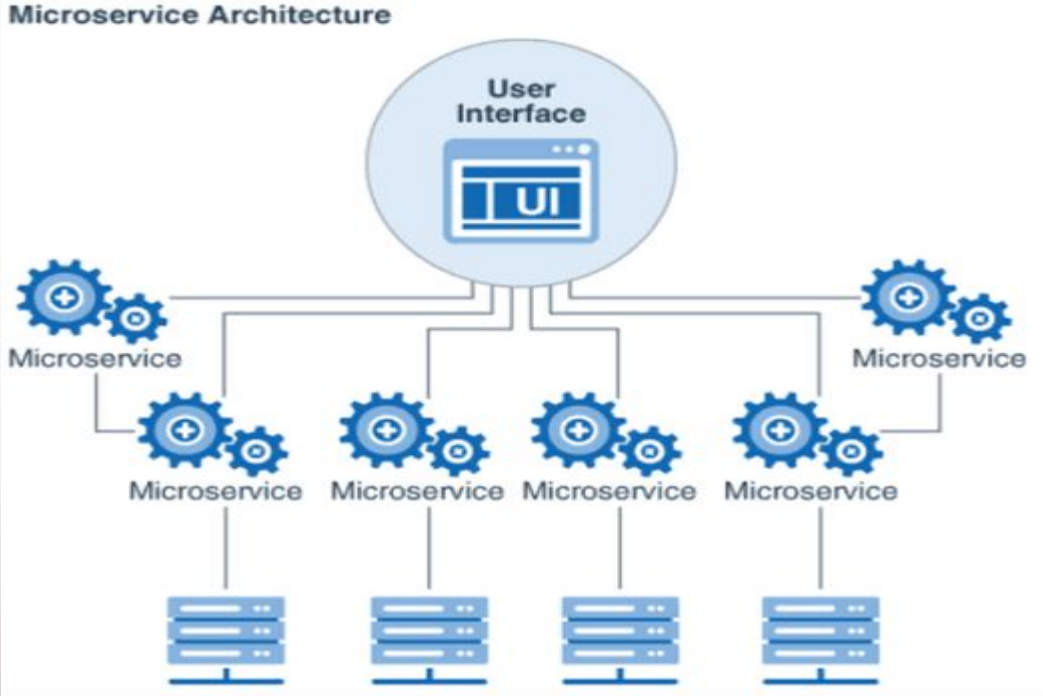
Задачи выпускной квалификационной работы:

1. Изучение теоретических и практических аспектов автоматизированного тестирования, его значения для разработки программного обеспечения, уровней автоматизации и места тестирования API в общем процессе автоматизированного тестирования, инструментов для автоматизации тестирования API;
2. Исследование организации, в которой будет осуществляться внедрение проекта автоматизированного тестирования (изучение организационной структуры, основных бизнес-процессов);
3. Анализ проекта разработки приложения кредитного конвейера, на базе которого будет осуществляться внедрение автоматизированного тестирования (архитектура приложения, предлагаемые инструменты и средства автоматизированного тестирования);
4. Разработка и реализация автоматизации тестирования API:
 - составление плана автоматизации тестирования API;
 - выбор стратегии автоматизации на проекте кредитного конвейера;
 - разработка тестовых сценариев;
 - настройка рабочего окружения;
 - разработка тестовых скриптов;
 - проектирование и реализация алгоритма автоматизации процесса тестирования API;
5. Оценка результатов тестирования;
6. Оценка эффективности внедрения автоматизированного тестирования.

Основные аспекты автоматизированного тестирования



Пирамида автоматизации



Микросервисная архитектура

Коммерческие инструменты



Бесплатные и условно-бесплатные инструменты



Инструменты собственной разработки



Инструменты для автоматизации тестирования API

Организационная структура ООО «БСЦ Мск»

Автоматизация тестирования проводится для компании ООО «БСЦ Мск»

Компания занимается разработкой и поддержкой банковского программного обеспечения

Тестированием занимается отдел обеспечения качества

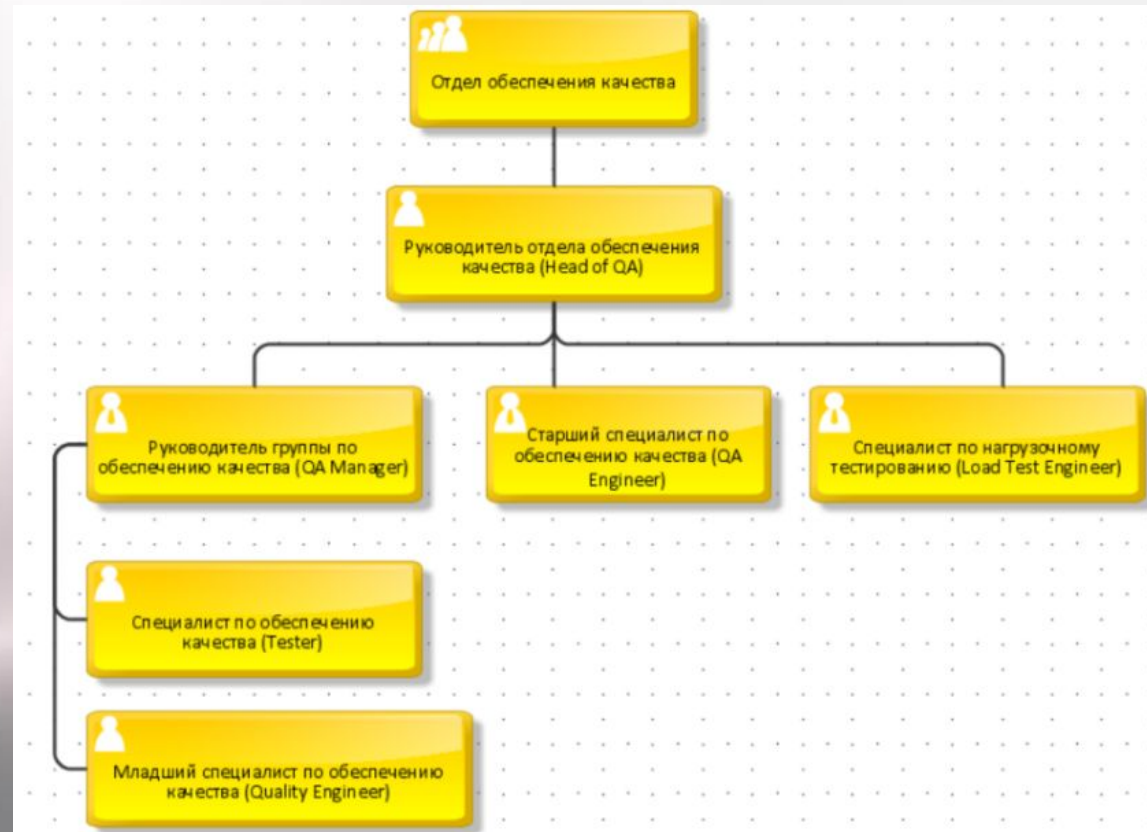
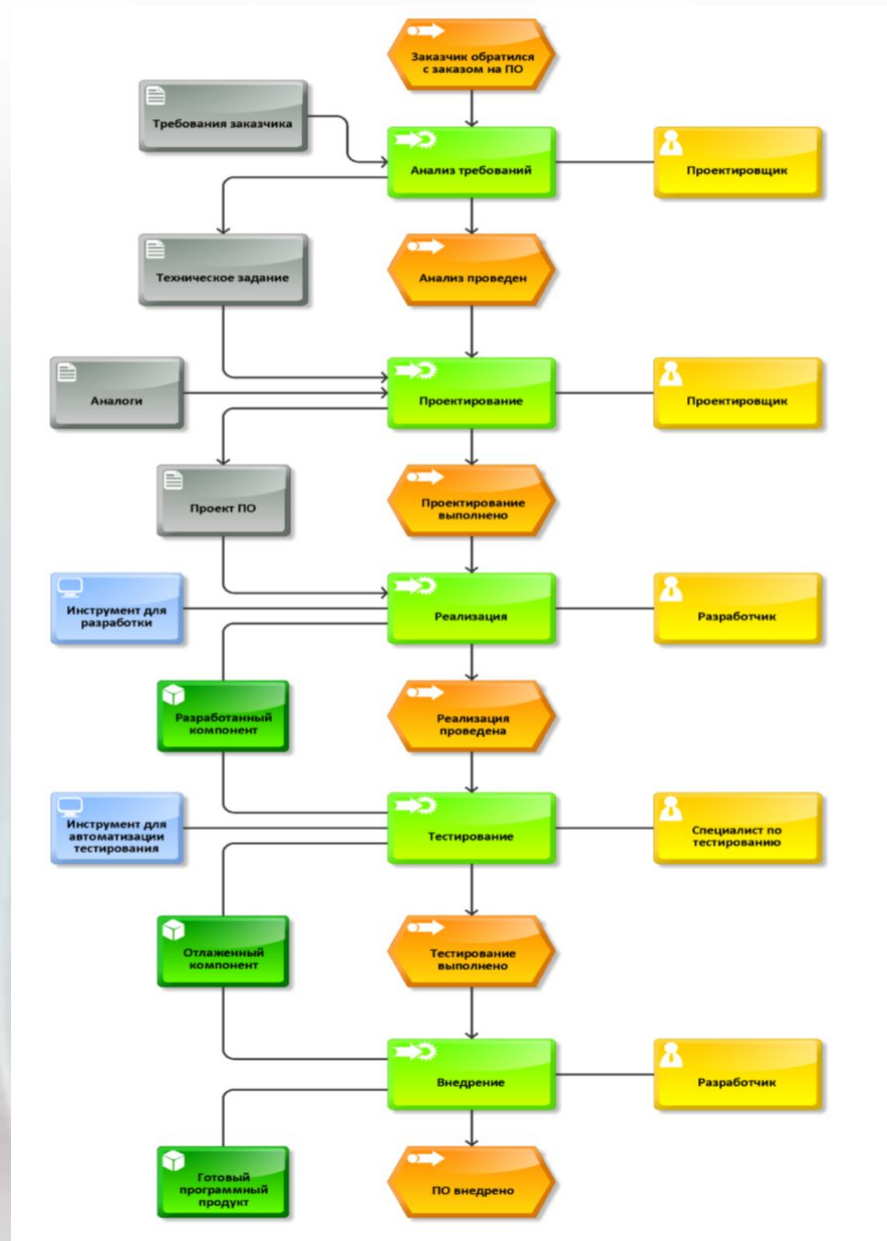


Схема процессов разработки ПО в нотации EPC

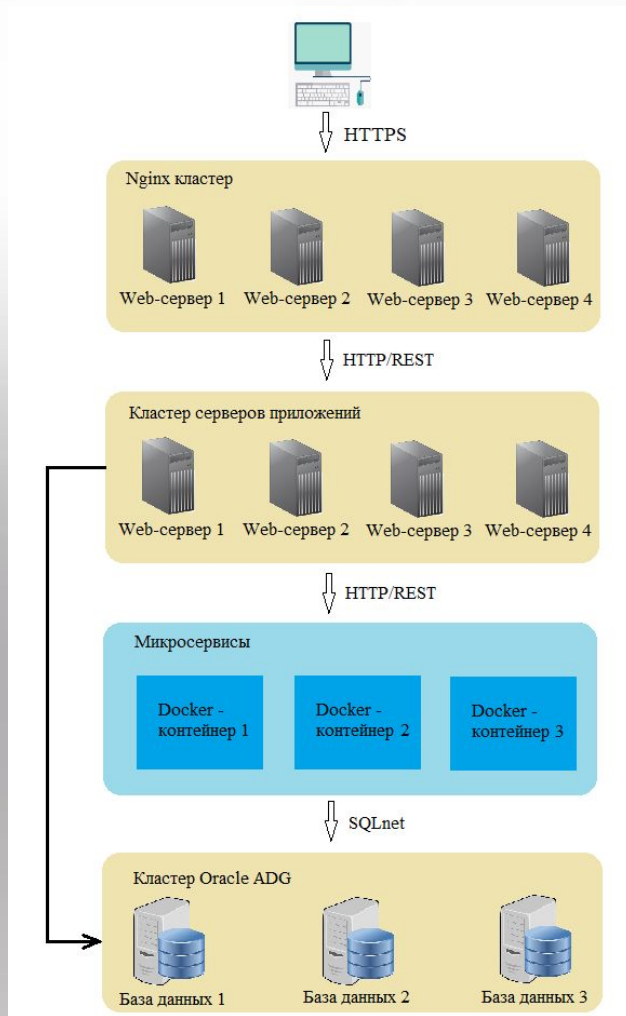


Нотация моделирования EPC (Event-driven Process Chain) ориентирована на построение алгоритмов взаимодействия в процессе выполнения конкретной работы.

Главные элементы:

- события, которые запускают или завершают работу;
- действия (работа), которая переводит систему из одного состояния в другое;
- исполнители работ;
- ресурсы и результаты работы (входы и выходы).

Архитектура тестируемого объекта

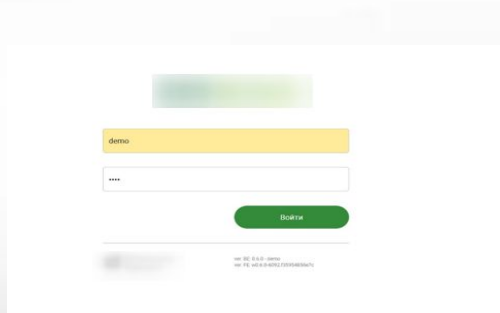


Техническая архитектура системы кредитного конвейера

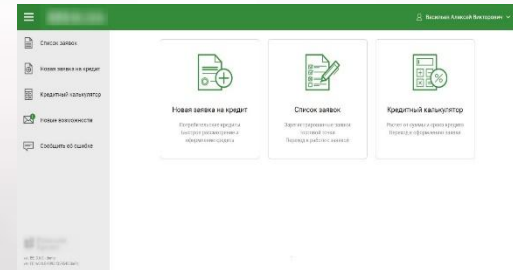


Бизнес-архитектура системы кредитного конвейера

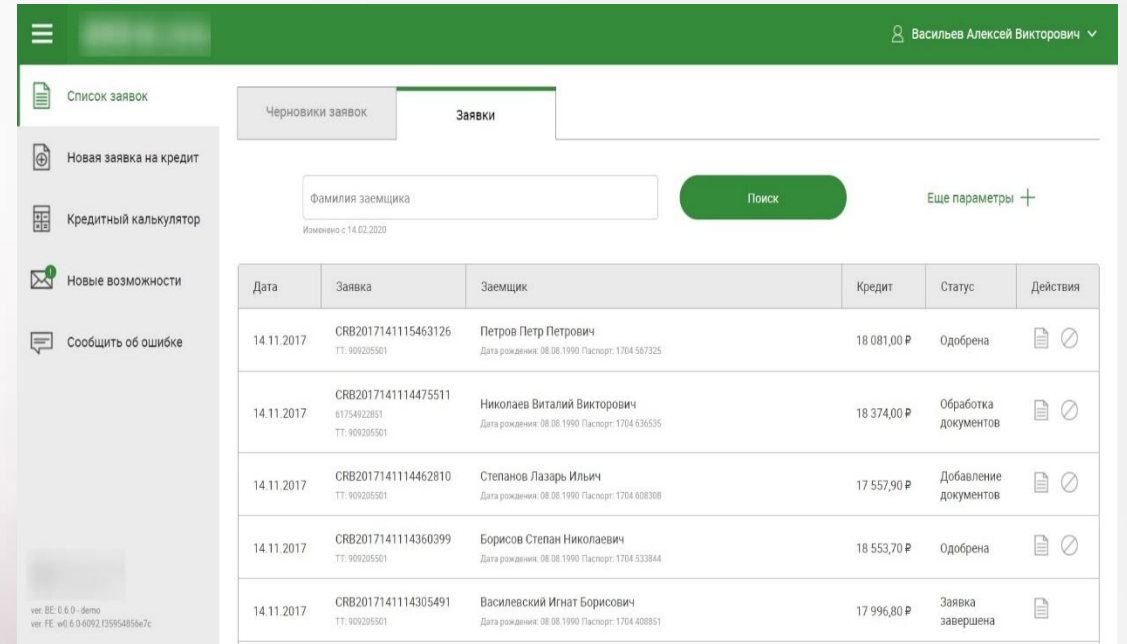
Интерфейс тестируемого объекта



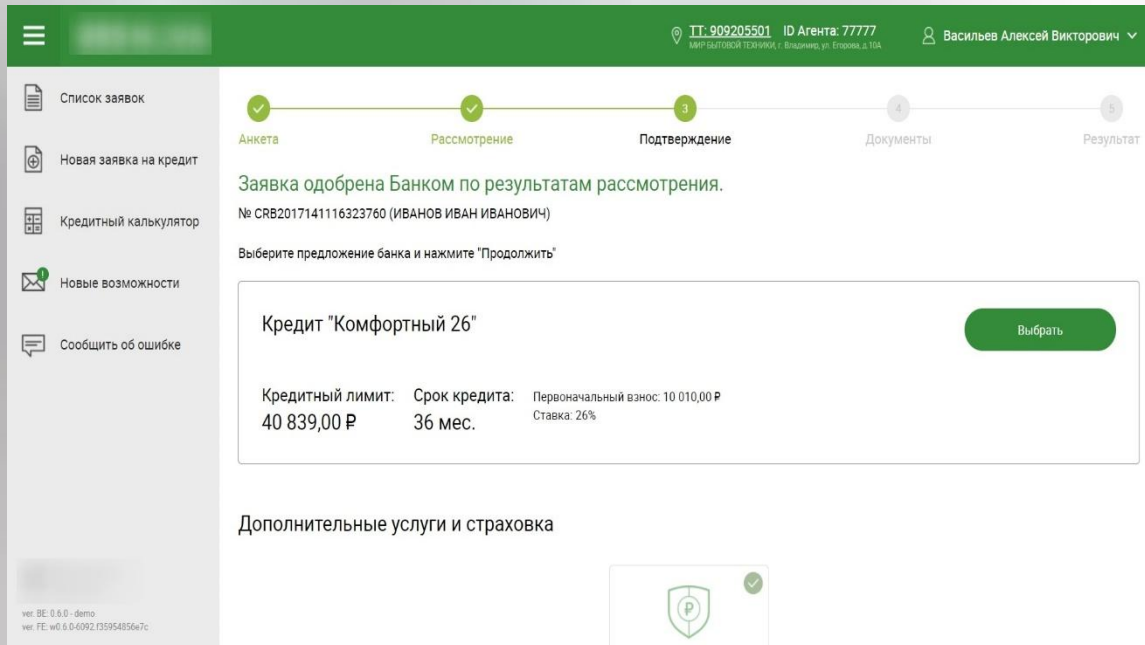
Форма авторизации



Главная страница



Список заявок



Создание новых заявок

Выбор инструментов автоматизации тестирования

Критерий	CTT *	Jmeter	Karate-DSL	KS **	Postman	REST-Assured	SoapUI	Tosca ***
Функционал отправки запроса HTTP	●	●	●	●	●	●	●	●
Эмуляция конечной системы REST, SOAP	●	●	●	●	●	●	●	●
Сравнение фактического ответа на запрос с ожидаемым	●	●	●	●	●	●	●	●
Freeware	●	●	●	●	●	●	●	●
Поддержка встраивания AT в CI/CD	●	●	●	●	●	●	●	●
Взаимодействие по протоколу HTTP	●	●	●	●	●	●	●	●
Взаимодействие по протоколу SOAP	●	●	●	●	●	●	●	●
Взаимодействие с MQ	●	●	●	●	●	●	●	●
Поддержка JMS	●	●	●	●	●	●	●	●
Наличие UI	●	●	●	●	●	●	●	●
No coding	●	●	●	●	●	●	●	●
Работа с данными из БД	●	●	●	●	●	●	●	●

* CinimexTestTool ** Katalon Studio *** Tricentis Tosca

Сравнительный анализ инструментов тестирования API

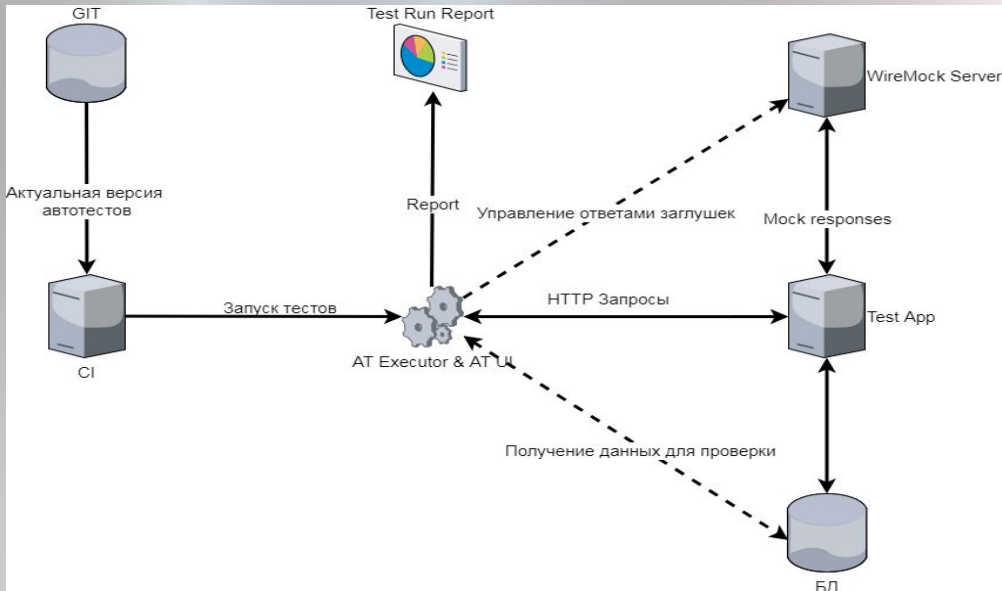
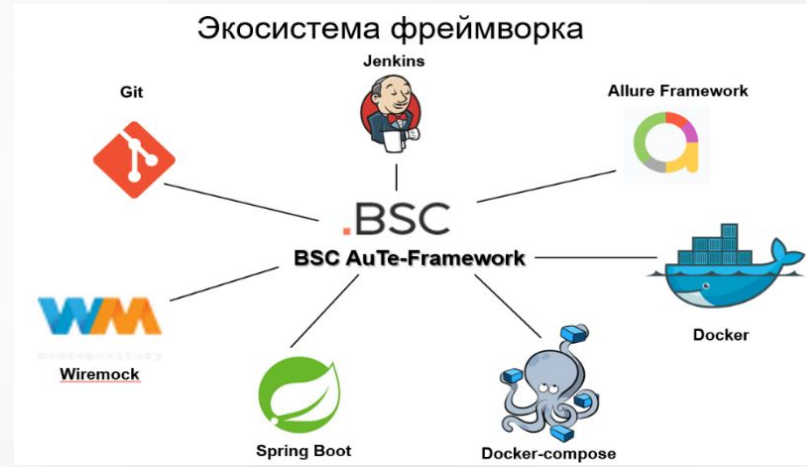


Схема интеграции



Экосистема BSC AuTe-Framework

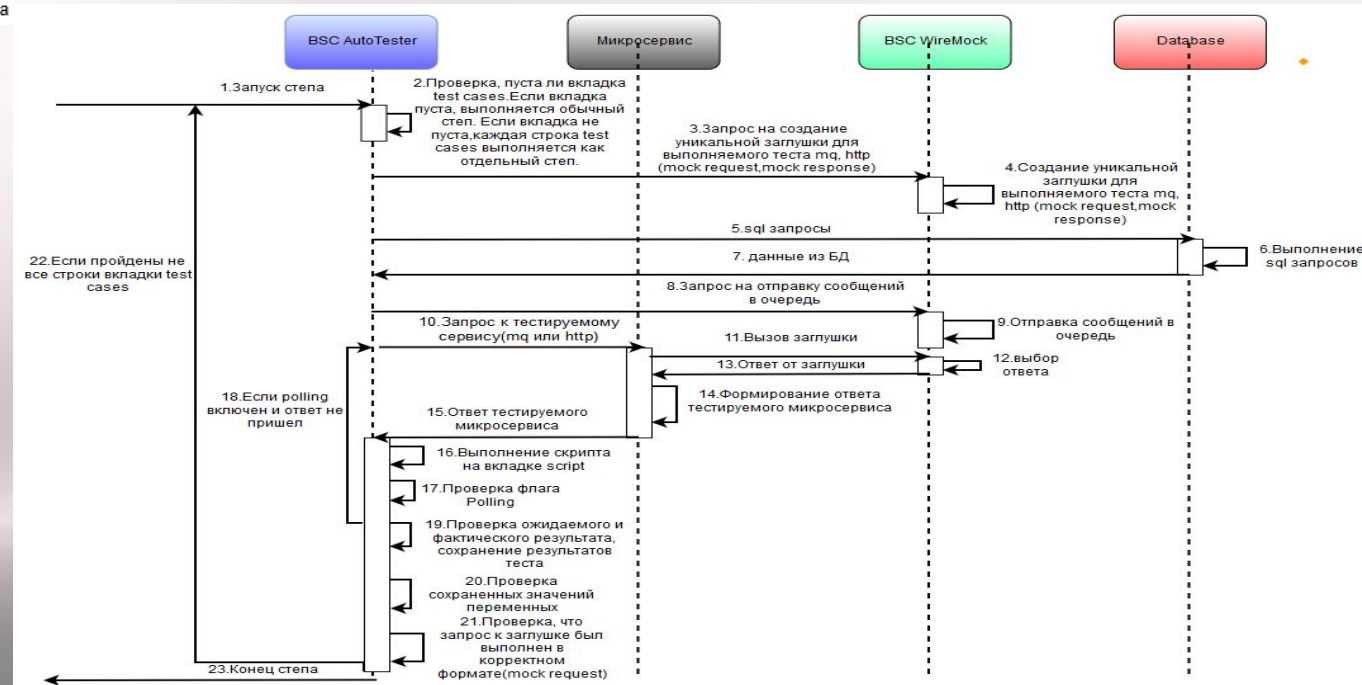
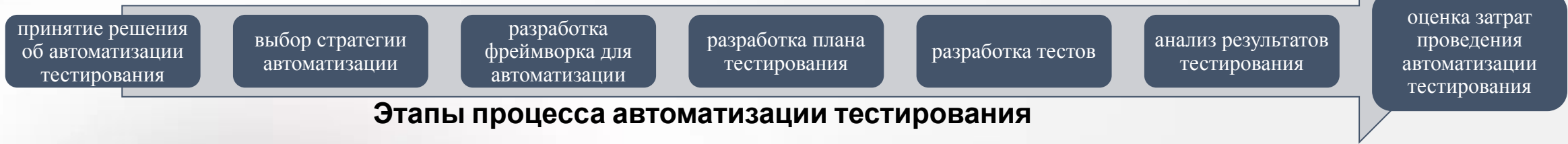
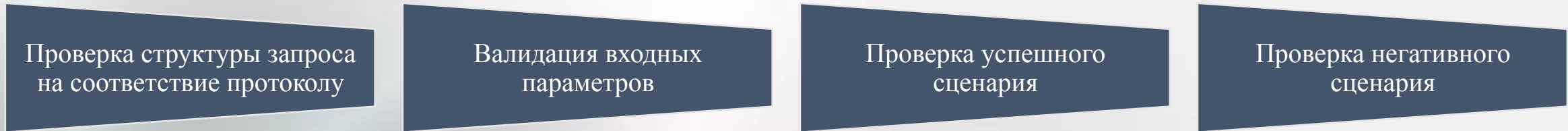


Диаграмма последовательности тестирования

Процесс автоматизации системы тестирования



Выбранная стратегия - «Operation Uranum»



Выбранные основные сценарии для проверки микросервиса на корректную работоспособность

Название	МкС ContractNumberGenerationCommand	
Функция	Микросервис для оркестрации обменов в рамках процесса генерации номеров счета и договора	
Действие	Ожидаемый результат	Результат теста: - Пройден - Провален
1. Обратиться к сервису со структурой запроса не соответствующей протоколу	HTTP status 400 code: «Некорректный протокол»	Пройден

Тестовый сценарий проверки структуры запроса на соответствие

Название	МкС ContractNumberGenerationCommand	
Функция	Микросервис для оркестрации обменов в рамках процесса генерации номеров счета и договора	
Действие	Ожидаемый результат	Результат теста: - Пройден - Провален
1. Обратиться к сервису, при этом значение name не соответствует ни одной из следующих команд: contractNumberGeneration	HTTP status 400 code: «Некорректный параметр»	Пройден
2. Обратиться к сервису, при этом отсутствуют следующие обязательные значения: Name, applicationID, additionalProperties.value, additionalProperties.name	HTTP status 400 code: «Отсутствует параметр»	Пройден

Тестовый сценарий проверки значений атрибутов структуры

Тестовые сценарии

Название	МкС ContractNumberGenerationCommand	
Функция	Микросервис для оркестрации обменов в рамках процесса генерации номеров счета и договора	
Действие	Ожидаемый результат	Результат теста: -Пройден -Провален
1.Обратиться к сервису, при этом значение name= contractNumberGeneration	Выполняется чтение кредитной заявки (выполняется субсценарий)	Пройден
2.Проверить обновление статуса заявки	Выполняется обновление статуса заявки. <code=contractNumberGeneration>	Пройден
3.Проверить, что каждого из продуктов выполняется следующий набор действий, описанных в шагах 4, 5, 6, 7 и 8	Выполняются шаги 4, 5, 6, 7 и 8	Пройден
4.Проверить определение параметров продукта	Выполняется субсценарий определения	Пройден
5.Проверить получение номеров счета и договора	Выполняется субсценарий получения	Пройден
6.Проверить выполнение расчет ПСК в % в руб	Выполняется субсценарий расчета	Пройден
7.Проверить выполнение расчета ПСК макс и ПСК мин, если класс продукта равен = 'CC' или 'DC'	Выполняется субсценарий расчета	Пройден
8.Проверить обновление заявки	Выполняется субсценарий обновления заявки	Пройден
9.Проверить обновление статуса заявки	Выполняется обновление статуса заявки. <code=ContractNumberGeneration.Done>	Пройден
10.Проверить выполнение запроса и обработки вектора решений	HTTP status 200 Выполняется вызов вектора решений	Пройден

Название	МкС ContractNumberGenerationCommand	
Функция	Микросервис для оркестрации обменов в рамках процесса генерации номеров счета и договора	
Действие	Ожидаемый результат	Результат теста: -Пройден -Провален
1.Проверить обновление статуса заявки, при этом получен ответ HTTP status !=200	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
2.Проверить поведение, если в ответе не получены продукты или HTTP status != 200	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
3.Проверить поведение, если в ответе не получены номера счета и договора или HTTP status != 200	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
4.Проверить поведение, если в ответе не получен расчет ПСК в %, в руб или HTTP status != 200	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
5. Проверить выполнение расчета ПСК макс и ПСК мин, если класс продукта != ('CC' или 'DC') или HTTP status != 200	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
6.Проверить поведение, если некорректно выполнен расчет ПСК макс и ПСК мин, если класс продукта равен = 'CC' или 'DC'	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
7.Проверить поведение, если в ответе получен HTTP status !=200 при обновлении заявки	Выполняется обновление статуса заявки. <code=contractNumberGeneration.Error>	Пройден
8. Проверить поведение, если в ответе получен HTTP status !=200 при обновлении статуса заявки	Выполняется обновление статуса заявки. <code=ContractNumberGeneration.Error>	Пройден
9.Проверить поведение, если в ответе вернулся HTTP status !=200 при выполнении запроса и обработки вектора решений	Вызов не происходит. Основной сценарий завершается	Пройден

Тестовые скрипты

The screenshot shows the BSC AutoTester interface. At the top, there's a navigation bar with 'BSC AutoTester', 'Проекты', and 'Help'. Below it, a breadcrumb trail shows 'Проекты / ContractNumberGenerationCommand / 400-ec-badrequest / 400 EC-BadRequest'. The main area displays a test script configuration for '400 EC-BadRequest'. The script is a POST request to '/command'. The request body is a JSON object:

```
{ "1234" }
```

. The expected status is 400. The response body is a JSON object:

```
{ "code": "protocol Incorrect" }
```

. The interface includes fields for 'Timeout', 'Repeat before each request', and 'Repeat before each request'.

Проверка структуры запроса на соответствие протоколу

The screenshot shows the BSC AutoTester interface for a test script named '400 EC-Validation'. The script is a POST request to '/command'. The request body is a JSON object:

```
{ "applicationId": "2019091200002891" }
```

. The expected status is 400. The response body is a JSON object:

```
{ "code": "validation.Error", "titleErrors": [ { "attr": "name", "code": "validation.Missing" } ] }
```

. The interface includes fields for 'Timeout', 'Repeat before each request', and 'Repeat before each request'.

Проверка на валидацию обязательного поля

The screenshot shows the BSC AutoTester interface for a test script named '200 OK'. The script is a POST request to '/command'. The request body is a JSON object:

```
{ "applicationId": "2019091200002891", "name": "contractNumberGeneration" }
```

. The expected status is 200. The response body is empty. The interface includes fields for 'Timeout', 'Repeat before each request', and 'Repeat before each request'.

Сценарий 200 ОК

The screenshot shows the BSC WireMock interface. At the top, there's a navigation bar with 'BSC WireMock', 'Mappings', 'JMS Mappings', 'REST log', and 'MQ log'. Below it, there's a search bar and a table of requests. The table has columns for 'Time', 'Method', 'Uri', and 'Response code'. The table contains 16 rows of request logs.

Time	Method	Uri	Response code
2020-04-04T00:45:30Z	POST	http://stub:8080/bsc-wire-mock/contract_creation/v3/command	200
2020-04-04T00:45:30Z	POST	http://stub:8080/bsc-wire-mock/auth/realm/arsnova/protocol/openid-connect/token	200
2020-04-04T00:45:30Z	POST	http://stub:8080/bsc-wire-mock/loan_application_events/v1/commands	200
2020-04-04T00:45:30Z	POST	http://stub:8080/bsc-wire-mock/params_collector/v2/decision	200
2020-04-04T00:45:30Z	POST	http://stub:8080/bsc-wire-mock/loan_application_events/v1/events	200
2020-04-04T00:45:30Z	POST	http://stub:8080/bsc-wire-mock/applications/v3/applications/2019091200002891/status?_withResult=false	200
2020-04-04T00:45:30Z	PUT	http://stub:8080/bsc-wire-mock/applications/v3/applications/2019091200002891/product_contents_result/73107?_withResult=false	200
2020-04-04T00:45:29Z	GET	http://stub:8080/bsc-wire-mock/loan_product_irrv1/loan_product_irrv1/productCodes=RCCFMCC_INST	200
2020-04-04T00:45:29Z	POST	http://stub:8080/bsc-wire-mock/loan_product_irrv1/check	200
2020-04-04T00:45:29Z	POST	http://stub:8080/bsc-wire-mock/contract_and_account_numbers_generator/v1/numbers_generations?mode=0&clientId=1041302707&resident=true&operationCode=348¤cy=810	200
2020-04-04T00:45:29Z	POST	http://stub:8080/bsc-wire-mock/auth/realm/arsnova/protocol/openid-connect/token	200
2020-04-04T00:45:29Z	POST	http://stub:8080/bsc-wire-mock/auth/realm/arsnova/protocol/openid-connect/token	200
2020-04-04T00:45:29Z	GET	http://stub:8080/bsc-wire-mock/product/v3/products?productClass=CC&tag=Direct&productCodes=RCCFMCC_INST&_fields=financeOperation.currency	200
2020-04-04T00:45:28Z	POST	http://stub:8080/bsc-wire-mock/applications/v3/applications/2019091200002891/status?_withResult=false	200
2020-04-04T00:45:28Z	GET	http://stub:8080/bsc-wire-mock/applications/v3/applications/2019091200002891?_expand=borrower/productContentsResult.statusHistory	200

BSC-WireMock

Проблемы и решения при создании автотестов

При каждом новом запуске сценариев автоматически генерируется идентификатор запроса, передаваемый во внешние сервисы, поэтому использование написанных ранее автотестов невозможно.

- Для решения данной проблемы применялось ключевое слово `*ignore*`. С его помощью сравнение запросов выполняется с возможностью игнорирования передаваемого параметра

В рамках одного сценария при создании шагов необходимо учесть, что некоторые вызовы к заглушкам кэшируются.

- Если запустить автотесты параллельно, не все вызовы, которые мы ожидаем, могут отправиться во внешние системы. Из-за данной причины некоторые шаги могли упасть с ошибкой «Timeout for requests for stubs has expired». Для того, чтобы решить данную проблему, необходимо настроить тайм-ауты и время ожидания запросов (в мс) между вызовами. Благодаря настройкам времени отсрочки запуска шага, кэшированные запросы являются не действительными и заглушки обрабатываются корректно.

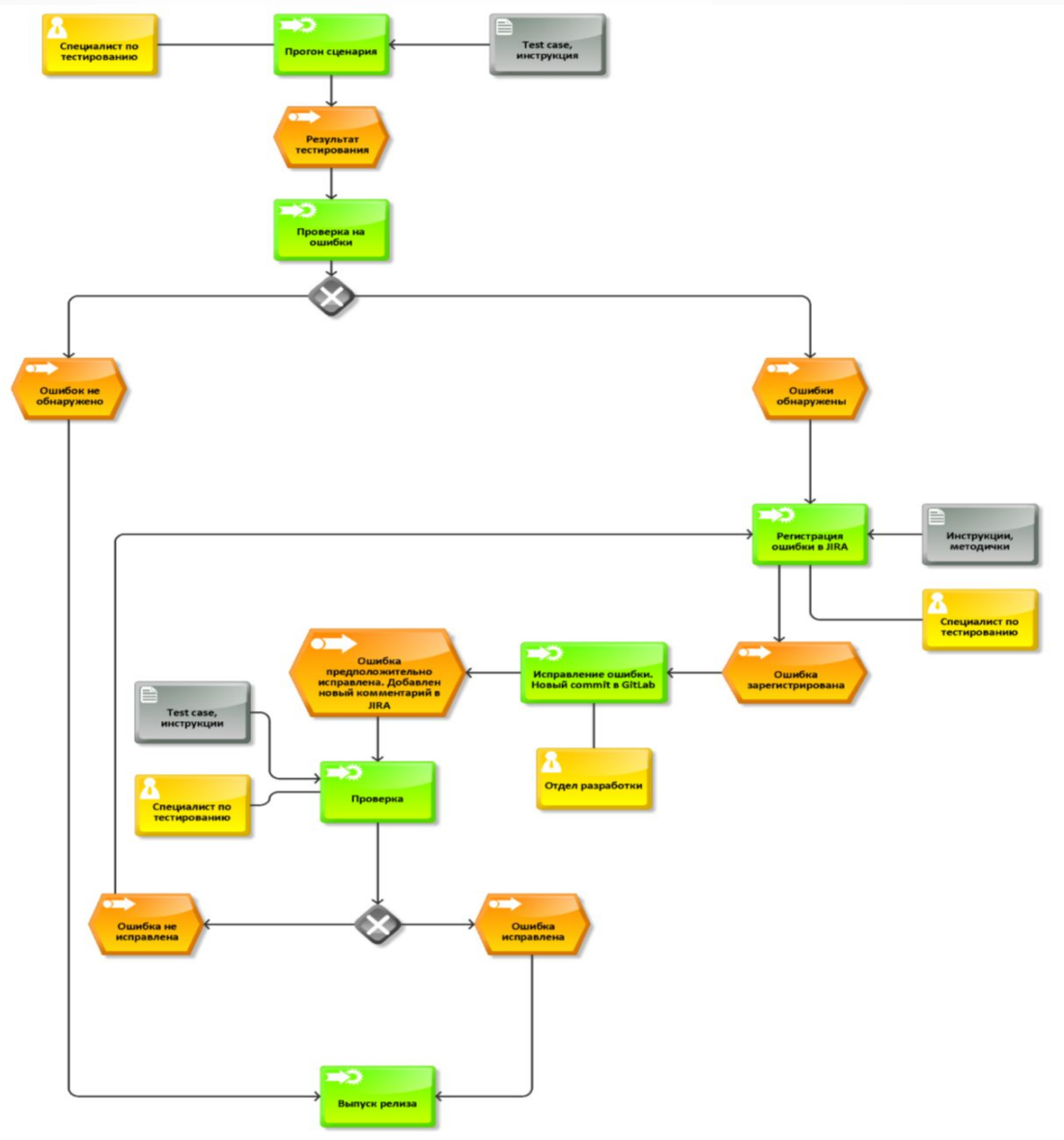
В рамках одного шага выполнялись POST-запросы с одинаковым URL, но с разными передаваемыми параметрами.

- Для того, чтобы автотестер различил их, необходимо настроить «Type matching» и «Matching». В поле «Type matching» выбирается признак «equalToJson», по которому будут сравниваться запросы. В поле «Matching» прописывается уникальное значение из ожидаемого запроса. После проделанных шагов автотестер в нужном порядке отправляет запросы во внешнюю систему.

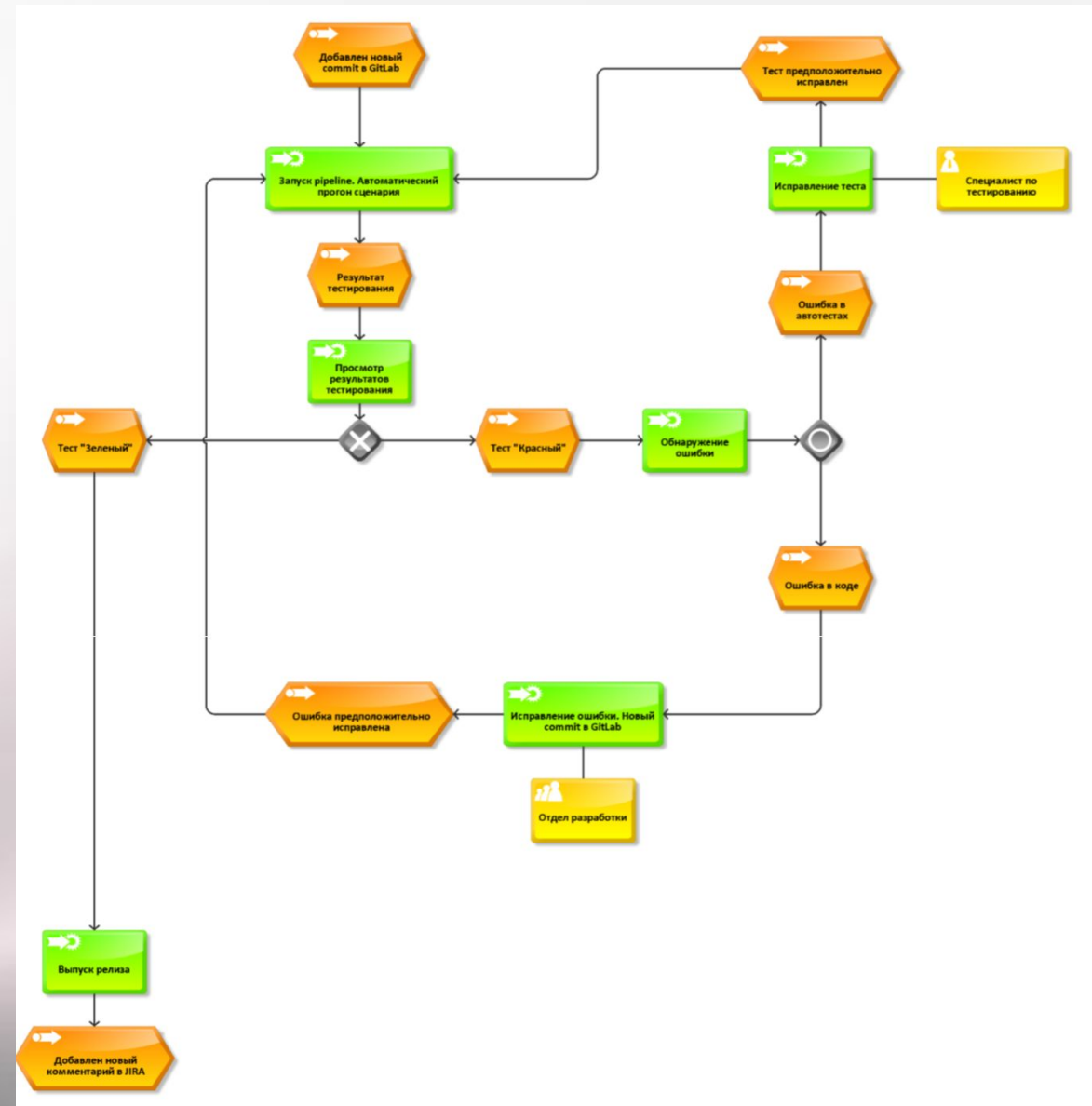
Тестируемый микросервис взаимодействует с тремя защищенными сервисами.

- Для того, чтобы проверить корректность вызовов, необходимы токены для аутентификации пользователей. Для их получения микросервис отправляет 3 одинаковых запроса по одному URL во внешнюю систему. Чтобы избежать дублирования заглушек, во вкладке «Ожидаемый запрос» проставляется repeat count равный трем. В ответах создаются 3 REST-заглушки с полученными токенами, с указанием порядка ответов от сервиса. Так как у любого токена есть «время жизни», в рамках автотестов необходимо настроить параметры на срок – 0 секунд. Данная манипуляция поможет избежать кэширования ответов от стороннего сервиса.

Проектирование алгоритма автоматизации процесса тестирования API

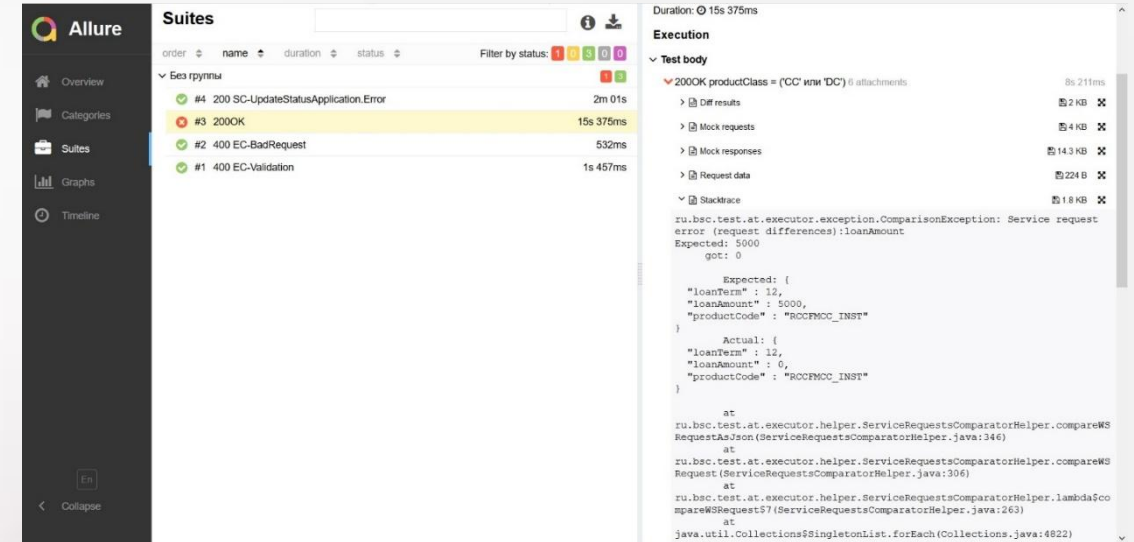
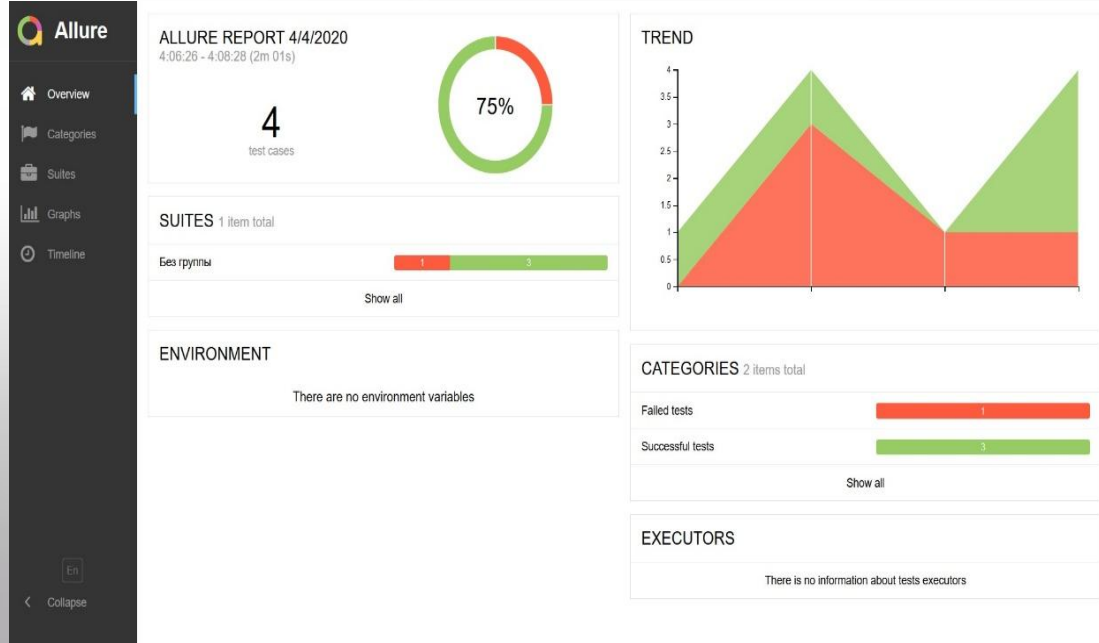


Алгоритм ручного тестирования (без использования инструментов автоматизации)



Разработанный алгоритм автотестирования

Результаты тестирования



Неуспешно выполненный тестовый сценарий

Блок «ALLURE REPORT»

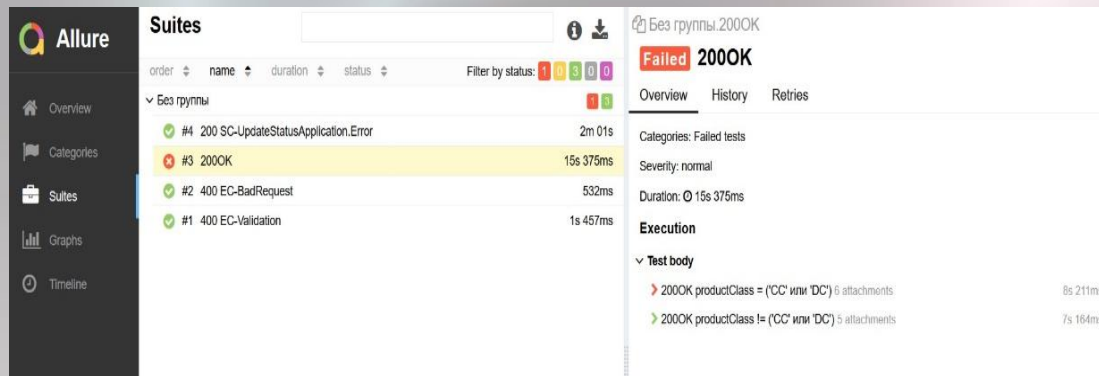
- включает в себя дату и время прохождения теста, общее количество пройденных тестов, а также диаграмму с указанием процента и количества успешных и упавших в процессе выполнения тестов

Блок «SUITES»

- показывает распределение результатов тестов по тестовым наборам

Блок «TREND»

- показывает прохождения тестов от сборки к сборке



Allure-отчет

Оценка эффективности внедрения автоматизированного тестирования

Расчет выгоды внедрения автоматизации

$$K = \frac{N * T * P}{L + T' * P'}$$

N – количество версий микросервиса, которые планируется выпустить в ходе реализации проекта;

T – примерное время, затрачиваемое на ручное выполнение тест-кейсов для одного микросервиса специалистом по тестированию;

P – зарплата специалиста по тестированию;

L – стоимость лицензии на средство автоматизации (так как расчет проводился для конкретного случая, в котором ПО для создания тестовых скриптов было уже разработано в компании, то значение данного показателя было принято за 0);

T' – примерное время на разработку, поддержку, выполнение автоматических тестов для одного микросервиса специалистом по автоматизации тестирования;

P' – зарплата специалиста по АТ

$$K = \frac{2 * 1440 * 25000}{0 + 960 * 30000} \approx 3$$

Расчет эффективности автоматизации

$$AT_{cr} + AT_{errval} + AT_{upd} < MT_{val} * N$$

AT_{cr} – время на создание автоматического теста;

AT_{errval} – среднее время на понимание причины «падения» автоматического теста;

AT_{upd} – среднее время на обновление автоматического теста;

MT_{val} – среднее время проведения ручного теста;

N – количество выполнений в течение одной итерации

$$15 + 15 + 10 < 20 * 4$$

$$40 < 80$$

Вывод:

K > 1, следовательно, автоматизация тестирования выгодна для проекта.

Расчет эффективности автоматизации тестирования показывает, что автоматические тесты эффективно влияют на проект и их встраивание в непрерывную интеграцию является целесообразным решением.

Заключение

Результаты работы:

- 1) Достигнута цель разработки проекта автоматизированного тестирования программного обеспечения – кредитного конвейера, особенностью которого является собственный стек средств и инструментов автоматизации тестирования.
- 2) Разработан и реализован алгоритм автоматизации тестирования API (составлен план автоматизации, выбрана стратегия автоматизации, разработаны тестовые сценарии, изучена настройка рабочего окружения, разработаны тестовые скрипты, спроектирован и реализован алгоритм автоматизации тестирования).
- 3) Проведена оценка результатов тестирования и эффективности внедрения автоматизированного тестирования.
- 4) Выявлены преимущества и недостатки проекта автоматизации тестирования.

Преимущества проекта автоматизации тестирования:

- гибкость набора инструментов автоматизации тестирования (разработчики могут самостоятельно менять компоненты в зависимости от тестируемой версии программного продукта);
- экономия за счет средств на обучение и переквалификацию персонала в результате частичного применения инструментов, использованных в организации ранее;
- сокращение общего времени разработки кредитного конвейера за счет ускорения процесса поиска ошибок в приложении.

Практическая значимость ВКР: разработанные тесты в настоящий момент используются в процессе тестирования приложения кредитного конвейера, их внедрение позволило значительно сократить расходы компании на тестирование, ускорить время разработки и улучшить качество программного продукта.

**Спасибо за
внимание!**