

ДОКЛАД

Tools and Infrastructure

15.10 / 15:45 – 16:15 UTC+3 (Europe/Moscow) ЗАЛ 3

Автоматическая генерация тестов как битва с драконами

RU



Спикеры



Егор Куликов
Explyt



Илья Муравьев
Explyt

Эксперты



Андрей Кулешов
Positive Technologies



2018-2020



GPT-3



Ponicode

2020-2022



GPT-3.5



codeium

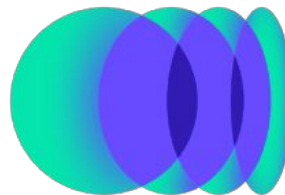
2022-2024



GPT-4o



deepseek
coder



Cursor

Новый подход к генерации тестов



Схема экспериментов с LLM



Модель: GPT-4o-2024-08-06



Температура: 0
(~детерминизм)



User ▾



Generate class with tests for `StringUtils`.

kotlin

Copy code

```
object StringUtils {  
    fun isCorrectParensSequence(str: String): Boolean {  
        var opened = 0  
  
        for (char in str) {  
            when (char) {  
                '(' -> opened++  
                ')' -> {  
                    if (opened <= 0) return false  
                    opened--  
                }  
                else -> return false  
            }  
        }  
  
        return opened == 0  
    }  
}
```



Почему не стоит недооценивать результат?

“Утилитных” классов больше, чем кажется!

- работа со строками, с файлами
- обёртки над популярными библиотеками
- преобразования Entity в DTO
- и др.



```
public ProductResponse createResponse(Product product) {

    ProductResponse productResponse = new ProductResponse();
    productResponse.setName(product.getName());
    productResponse.setUrl(product.getUrl());
    productResponse.setProductVariants(product.getProductVariantList()
        .stream()
        .map(variant -> ProductVariantDTO
            .builder()
            .id(variant.getId())
            .price(variant.getPrice())
            .thumb(variant.getThumb())
            .stock(variant.getStock())
            .color(ColorDTO
                .builder()
                .name(variant.getColor().getHex())
                .hex(variant.getColor().getName())
                .build())
            .build())
        .collect(Collectors.toList()));

    return productResponse;
}
```


Почему не стоит недооценивать результат?

```
34 @Test
35 public void testConvertProductToProductResponseWithSingleVariant() {
36     ...
66     assertEquals(expected: "Green", productVariantDT0.getColor().getName());
67     assertEquals(expected: "#00FF00", productVariantDT0.getColor().getHex());
68 }
```

Run ProductResponseConverterTest.testConvertProductToPro...



Tests failed: 1 of 1 test – 11 ms

ProductR 11 ms
testCc 11 ms

```
/home/ilya/.jdk/temurin-11.0.22/bin/java ...
```

org.opentest4j.AssertionFailedError:

Expected :Green

Actual :#00FF00

[<Click to see difference>](#)

Не разорительно ли это?



```
import org.junit.jupiter.api.Assertions.*
import org.junit.jupiter.api.Test

class StringUtilsTest {

    @Test
    fun `test empty string`() {
        assertTrue(StringUtils.isCorrectParensSequence(""))
    }

    @Test
    fun `test single pair of parentheses`() {
        assertTrue(StringUtils.isCorrectParensSequence("()"))
    }

    @Test
    fun `test multiple pairs of parentheses`() {
        assertTrue(StringUtils.isCorrectParensSequence("(()))"))
        assertTrue(StringUtils.isCorrectParensSequence("()()"))
        assertTrue(StringUtils.isCorrectParensSequence("(()())"))
    }
}
```




Вся ли рутина исчезла?

Уточнять промпты

Generate test class for StringUtils class.
Use JUnit 5.5.2 test framework.
Use Mockito 4.2.0 mocking framework.

Исправлять неточные результаты
(постобработка)

```
@Test
fun `test empty list`() {
    val result = StringUtils.isCorrectParensSequence("()")
    assertTrue(result, "Empty list should be a correct")
}
```

 Import function 'assertTrue'

Промптинг: на грани добра и зла

Нетривиальное, но важное:

When testing positive scenarios, it's very important to set ALL fields of ALL DTOs and entities (even if the field seems irrelevant).

Забавное, но не менее важное:

I have no fingers.

I have the placeholders trauma. NEVER use placeholders.

Демотивирующий пример

```
object StatusUtils {  
    fun ensureCanceled(status: Status) {  
        check(status == Status.CANCELED)  
    }  
}
```



Generate class with tests for `StatusUtils`.

```
object StatusUtils {  
    fun ensureCanceled(status: Status) {  
        check(status == Status.CANCELED)  
    }  
}
```



```
class StatusUtilsTest {  
    @Test  
    fun `ensureCanceled should not throw exception when status is CANCELED`() {  
        StatusUtils.ensureCanceled(Status.CANCELED)  
    }  
  
    @Test  
    fun `ensureCanceled should throw exception when status is not CANCELED`() {  
        assertThrows(IllegalStateException::class.java) {  
            StatusUtils.ensureCanceled(Status.PENDING)  
        }  
        assertThrows(IllegalStateException::class.java) {  
            StatusUtils.ensureCanceled(Status.IN_PROGRESS)  
        }  
        assertThrows(IllegalStateException::class.java) {  
            StatusUtils.ensureCanceled(Status.COMPLETED)  
        }  
    }  
}
```



Generate class with tests for `StatusUtils`.

```
enum class Status {  
    ACCEPTED, REJECTED, CANCELED,  
}  
  
object StatusUtils {  
    fun ensureCanceled(status: Status) {  
        check(status == Status.CANCELED)  
    }  
}
```



```
class StatusUtilsTest {  
    @Test  
    fun `ensureCanceled should not throw exception when status is CANCELED`() {  
        StatusUtils.ensureCanceled(Status.CANCELED)  
    }  
  
    @Test  
    fun `ensureCanceled should throw exception when status is not CANCELED`() {  
        assertThrows(IllegalStateException::class.java) {  
            StatusUtils.ensureCanceled(Status.ACCEPTED)  
        }  
        assertThrows(IllegalStateException::class.java) {  
            StatusUtils.ensureCanceled(Status.REJECTED)  
        }  
    }  
}
```


Реальный проект




Contributor Covenant v1.4 adopted

build passing


openssf best practices passing

Our Commitment




Quality Lecture Videos

High quality lecture videos with demonstrations and tutorials to reinforce your understanding.



Experienced Instructors

Experienced instructors with years of experience in teaching and preparing students for exams.



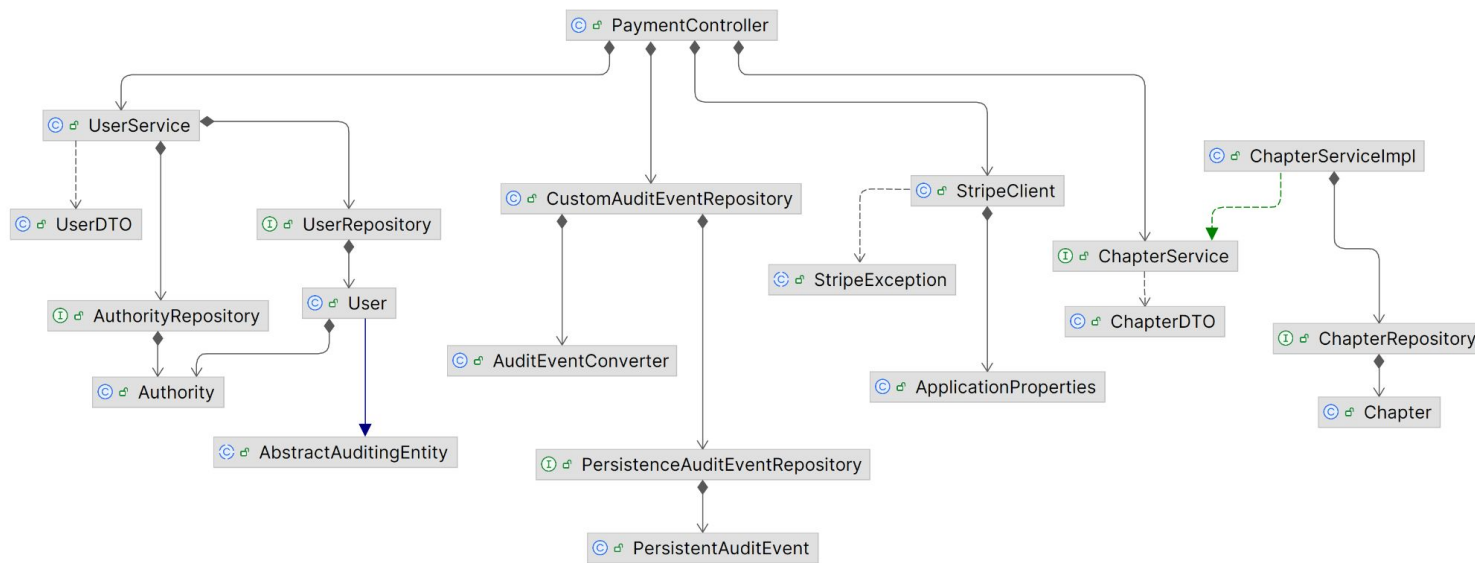
Revenue Neutral

Only show minimal ads in our videos to offset operational costs. One day we might even go Ad-free!

Тестируем PaymentController

Специфичные моменты:

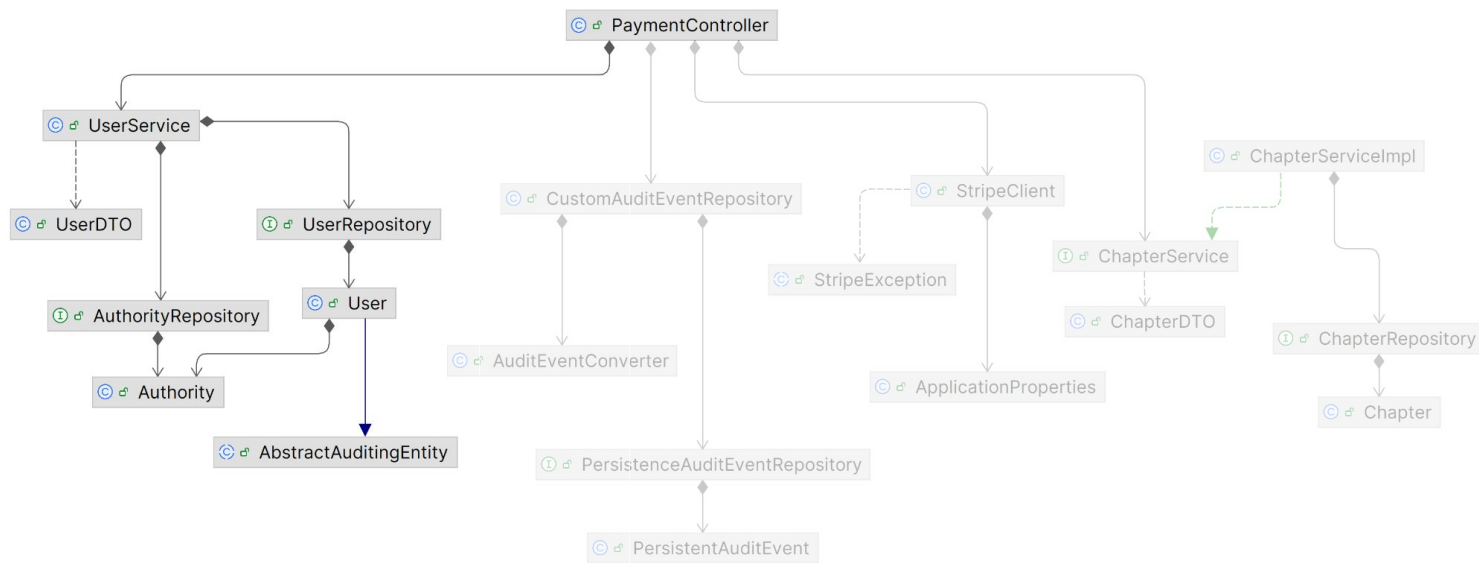
- объёмная и разветвлённая логика
- происходит взаимодействие с внешним сервисом
- используется Spring Security



Тестируем PaymentController

Специфичные моменты:

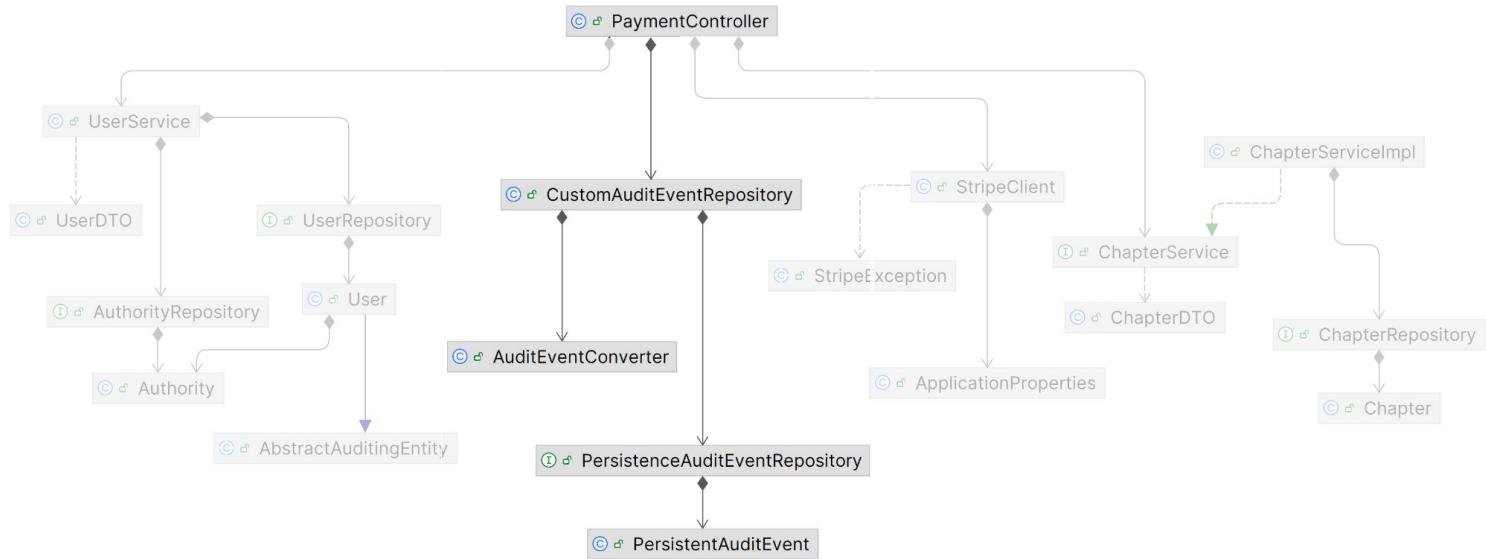
- объёмная и разветвлённая логика
- происходит взаимодействие с внешним сервисом
- используется Spring Security



Тестируем PaymentController

Специфичные моменты:

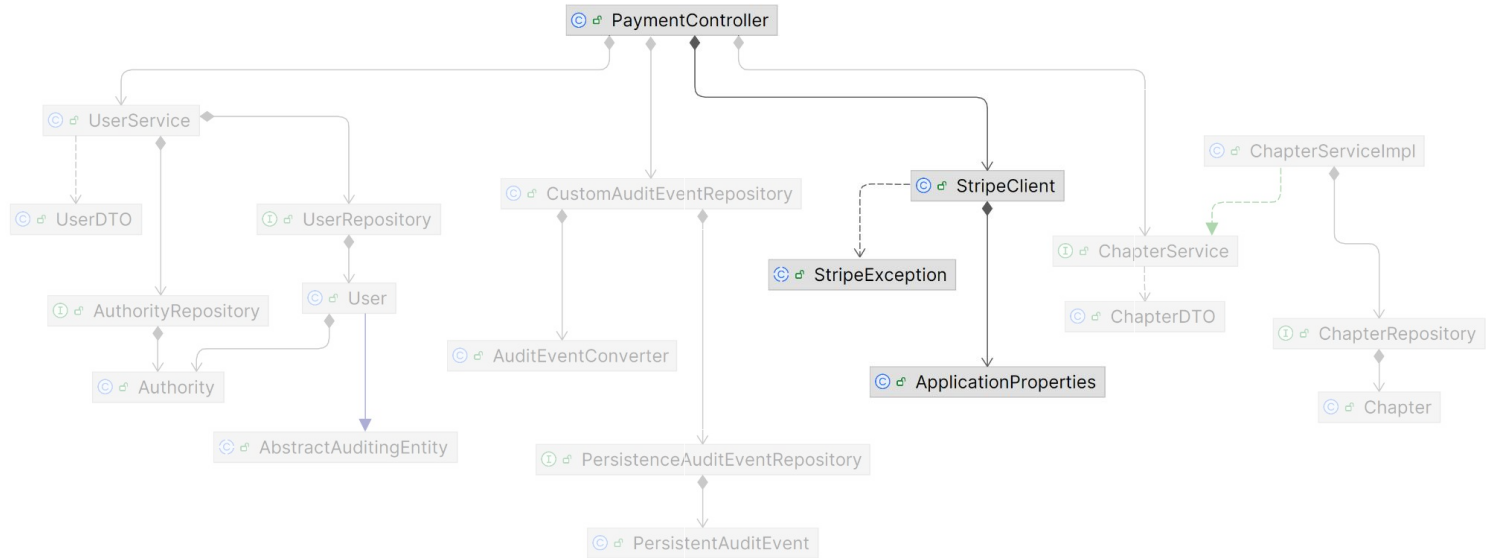
- объёмная и разветвлённая логика
- происходит взаимодействие с внешним сервисом
- используется Spring Security



Тестируем PaymentController

Специфичные моменты:

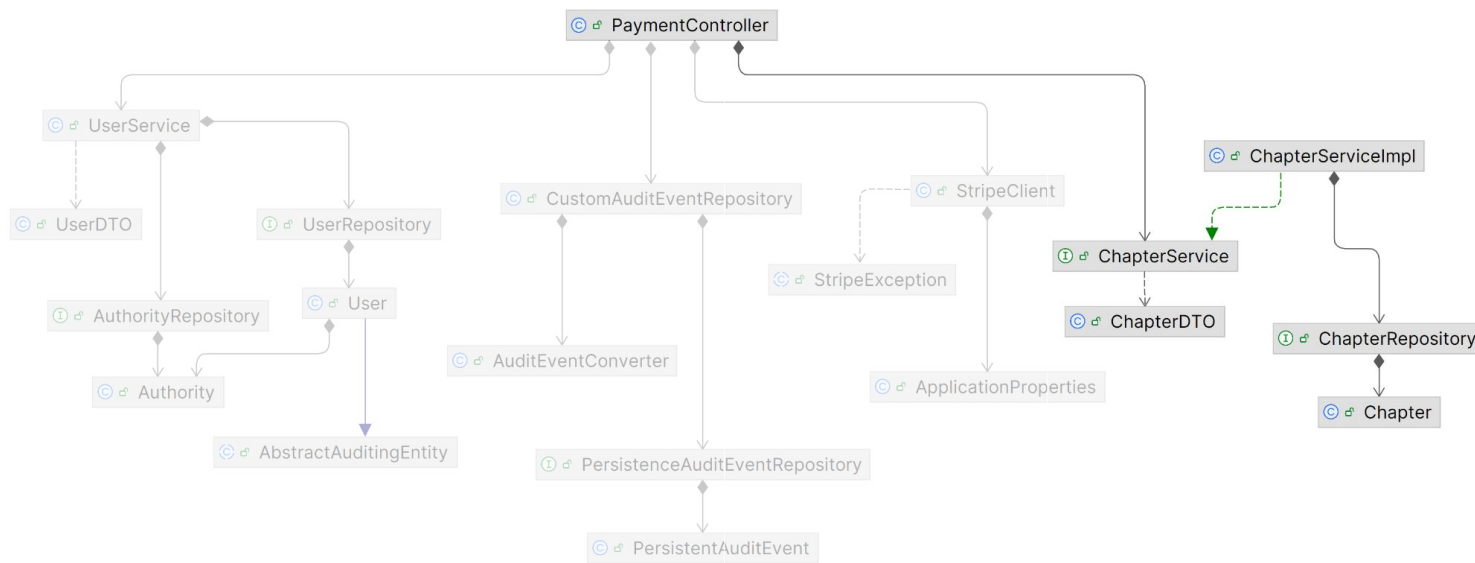
- объёмная и разветвлённая логика
- происходит взаимодействие с внешним сервисом
- используется Spring Security



Тестируем PaymentController

Специфичные моменты:

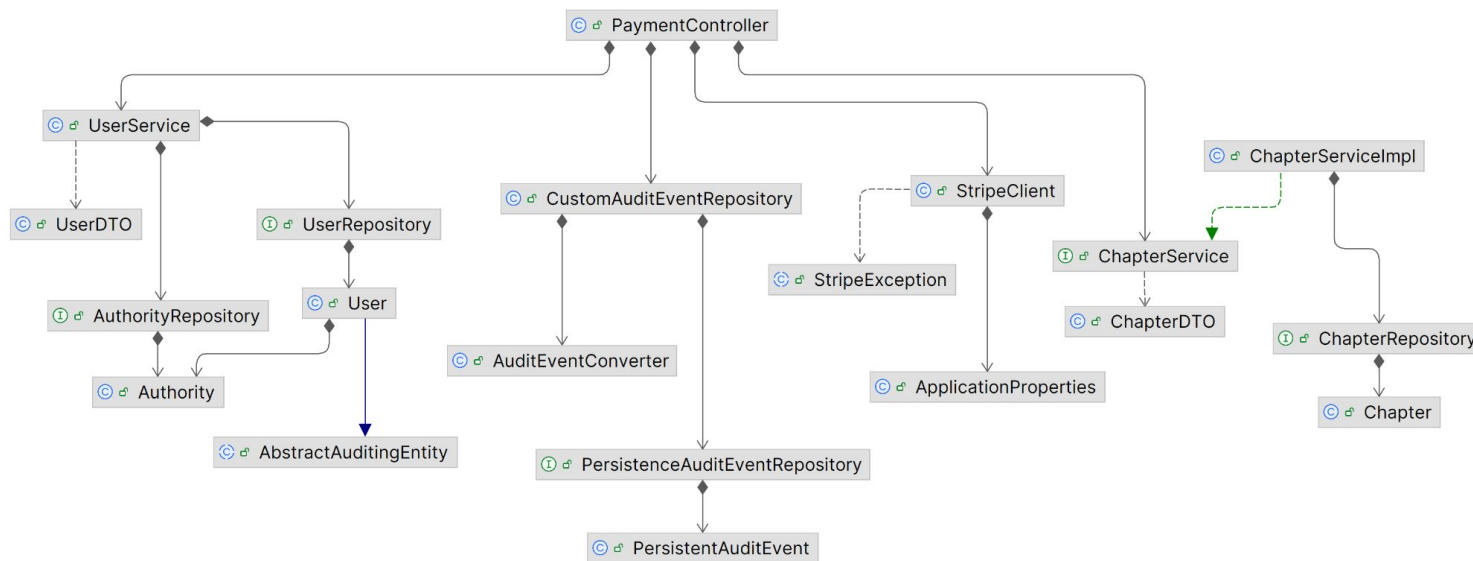
- объёмная и разветвлённая логика
- происходит взаимодействие с внешним сервисом
- используется Spring Security



Тестируем PaymentController

Специфичные моменты:

- объёмная и разветвлённая логика
- происходит взаимодействие с внешним сервисом
- используется Spring Security



Сбор контекста

- Трудоёмкая и зачастую довольно сложная задача
- Контекст свой для каждого объекта тестирования
- Существуют ограничения языковых моделей



ПРОСТИТЕ МЕНЯ, ЗА ВСЕ ЧТО Я СДЕЛАЛ



ДАЛЬШЕ БУДЕТ НАМНОГО ХУЖЕ



User ▾



INSTRUCTIONS


Generate tests for `PaymentController.chargeCard()` method which test the following scenarios. NEVER use placeholders or comments in the code. Your answer must include: complete source code of test class with source code of implemented methods.

Output format Instructions

Output test class (with ALL appropriate annotations) must be wrapped with backticks `java` and `.`

<TEMPLATE>

kotlin

 Copy code

```
*complete test class code without comments or placeholders*
```

Use JUnit 5.5.2 test framework.

Use Mockito 3.1.0 mocking framework.

When testing positive scenarios it's very important to set ALL fields of ALL DTOs and entities (even if the field seems irrelevant).



Тесты получены. Но насколько они хороши?



Типы граблей:

- ошибки компиляции
- тесты “не по шаблону”
- ошибки времени выполнения

Ошибки компиляции

Quick-fixes: **бесплатно**, но не всегда работает

```
fun SourceFolder.getModifiableContentEntry(): ContentEntry? = Egor Kulikov
    ModuleRootManager.getInstance(contentEntry.rootModel.module)
        .modifiableModel
        .contentEntries
        .find { entry -> entry.url == url }
```

Import class... com.intellij.openapi.roots.ContentEntry? ↵

Ошибки компиляции

Quick-fixes: **бесплатно**, но не всегда работает

```
fun SourceFolder.getModifiableContentEntry(): ContentEntry? = Egor Kulikov
    ModuleRootManager.getInstance(contentEntry.rootModel.module)
        .modifiableModel
        .contentEntries
        .find { entry -> entry.url == url }
```

Import class... com.intellij.openapi.roots.ContentEntry? ↵↵

Иногда этот подход даже **вредит**:

```
@Test
fun testEnsureCanceled_withNonCanceledStatus() {
    assertThrows(IllegalStateException::class.java) {
        StatusUtils.ensureCanceled(Status.ACTIVE)
    }
}
```

! Create enum constant 'ACTIVE'

! Create member property 'Status.Companion.ACTIVE'

Ошибки компиляции

LLM-fixes: дополнительный запрос



```
<...> // prompt & context

class PaymentControllerTest {
  <...>
  // The following line contains a compilation error that you must fix:
  // Cannot resolve method 'contains' in 'Object'
      event.getData().get("message").contains("chapter1") &&
  <...>
}
```



```
class PaymentControllerTest {
  <...>
      event.getData().get("message").toString().contains("chapter1") &&
  <...>
}
```

Статистические показатели

- 44 класса из 12 реальных проектов

Количество ошибок компиляции	GPT-4o-2024-08-06	DeepSeek-V2.5	llama-3.1-70b-versatile
Сгенерировано строк кода	8201	7165	7843
Код, сгенерированный LLM	653	423	1847
Quick-fixes	82	176	221
Quick-fixes + LLM-fixes	45	62	182

Тесты “не по шаблону”



```
@SpringBootTest
public class PaymentControllerTest {

    @InjectMocks
    private PaymentController paymentController;

    > <...>

    @Test
    public void testChargeCard_Success() throws <...> {
        // Arrange
    > <...>
        // Act
        Charge result = paymentController.chargeCard(request);
        // Assert
    > <...>
    }
}
```



прямой вызов
контроллера

Тесты “не по шаблону”



```
@SpringBootTest
public class PaymentControllerTest {

    @InjectMocks
    private PaymentController paymentController;

    > <...>

    @Test
    public void testChargeCard_Success() throws <...> {
        // Arrange
        <...>
        // Act
        Charge result = paymentController.chargeCard(request);
        // Assert
        <...>
    }
}
```

↑
прямой вызов
контроллера

Другие тесты в проекте

```
@AutoConfigureMockMvc
@SpringBootTest(classes = OpenLearnrApp.class)
public class UserJWTControllerIT {

    @Autowired
    private MockMvc mockMvc;

    > <...>

    @Test
    public void testAuthorize() throws Exception {
        // Arrange
        <...>
        // Act
        mockMvc.perform(post("/api/authenticate").<...>)
        // Assert
        <...>
    }
}
```

↑
тест через
mockMvc

Продолжаем дополнять контекст

Хорошие тесты:

- тесты для другого контроллера/сервиса/репозитория
- существующие тесты для этого класса

Процесс подбора примеров тоже можно автоматизировать.



+



=





User ▾



INSTRUCTIONS


Generate tests for `PaymentController.chargeCard()` method which test the following scenarios. NEVER use placeholders or comments in the code. Your answer must include: complete source code of test class with source code of implemented methods.

Output format Instructions

Output test class (with ALL appropriate annotations) must be wrapped with backticks `java` and `.`.

<TEMPLATE>

kotlin

 Copy code

```
*complete test class code without comments or placeholders*
```

</TEMPLATE>

Self-attention instructions

When generating tests for the given code, please assume that each new test will be included in the same test suite as the reference code.



Почему тест падает?

Падающий тест



Проблема в коде



Проблема в тестах

Типичные причины падения тестов

LLM “забыла”:

- про аутентификацию (наш случай)
- про сохранение сущностей в БД
- замокировать некоторые методы
- проставить некоторые филды

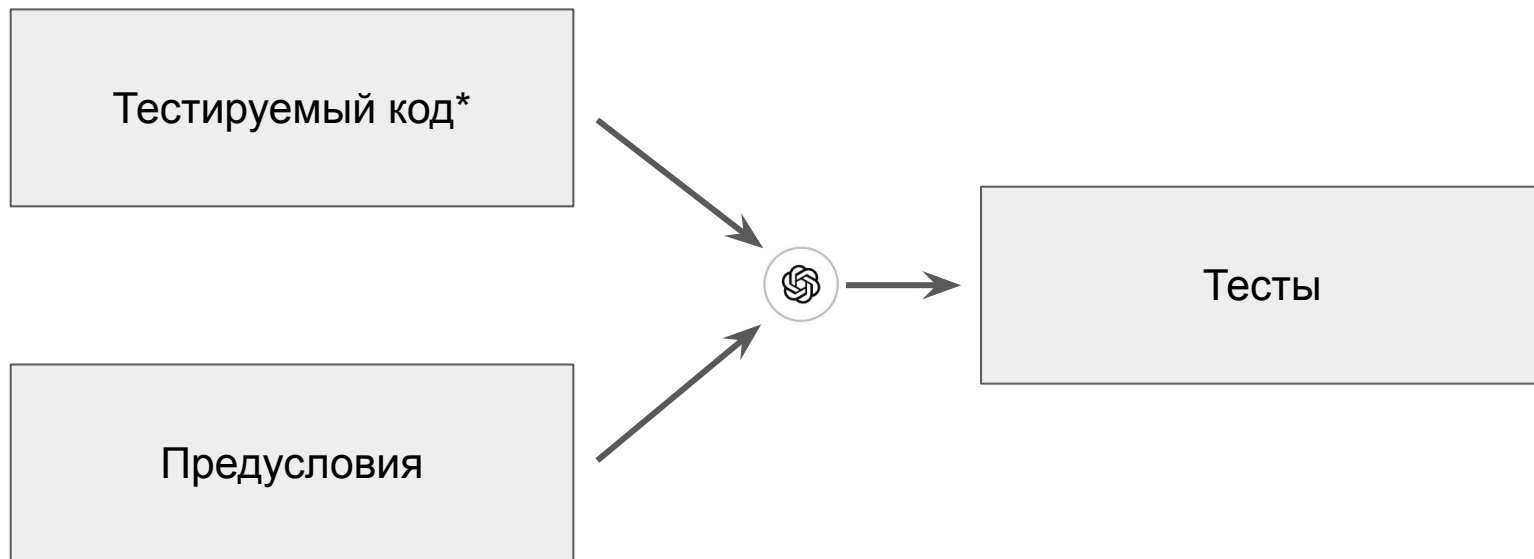
✘ PaymentControllerIT (com.asan 565 ms)
✔ testChargeCardUserNotLog: 451 ms
✘ testChargeCardSuccess() 96 ms
✘ testChargeCardStripeExcepti 18 ms

✘ Tests failed: 2, passed: 1 of 3 tests – 565 ms

```
Redirected URL = null  
Cookies = []
```

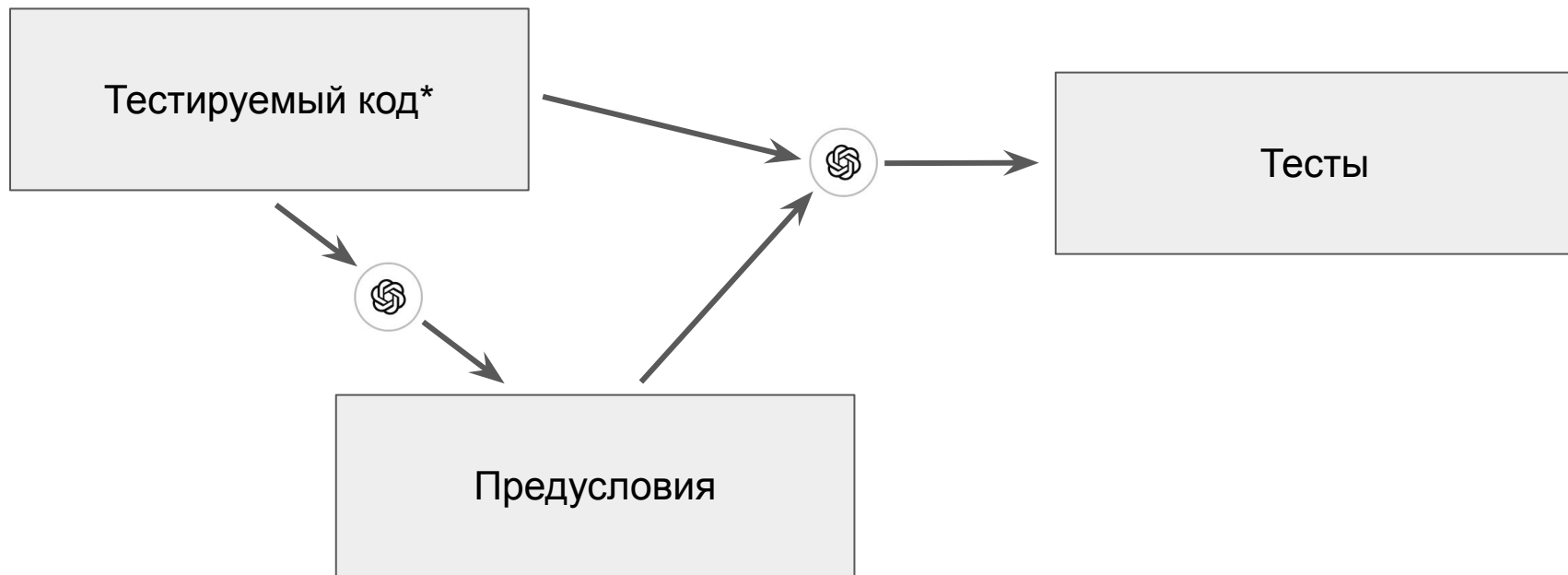
```
java.lang.AssertionError: Status expected:<200> but was:<401>  
Expected :200  
Actual   :401
```

Борьба с пропущенными условиями



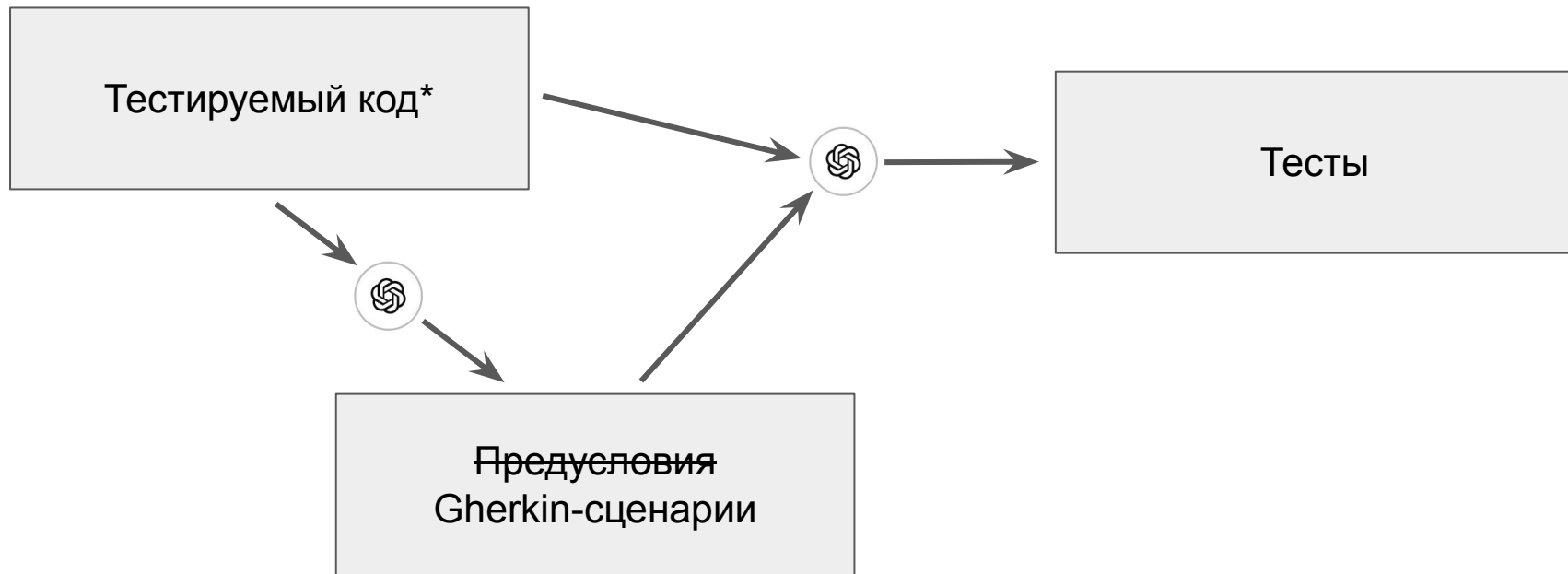
* включая контекст, примеры и список установленных библиотек :)

Борьба с пропущенными условиями



* включая контекст, примеры и список установленных библиотек :)

Борьба с пропущенными условиями



* включая контекст, примеры и список установленных библиотек :)

Gherkin-сценарии

Scenario: Payment failure due to invalid Stripe token

Given a user "test_user" is logged in

And "test_user" is saved in the system with the login "test_user"

And "test_user" has an invalid Stripe token "tok_invalid"

And chapters "chapter1" and "chapter2" are saved in the system with IDs "ch1" and "ch2"

And "test_user" has no current subscriptions

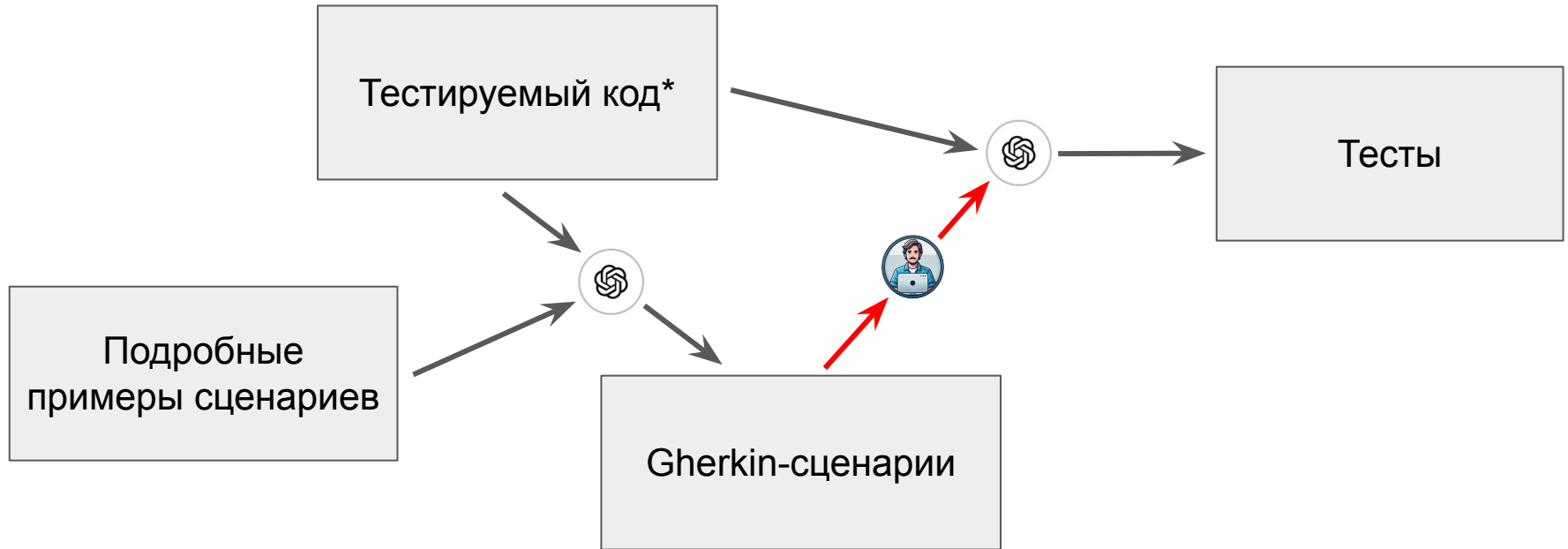
When "test_user" makes a payment with the token "tok_invalid" for chapters "ch1" and "ch2"

Then the payment should fail with a StripeCardError

And "test_user" should not be subscribed to any chapters

And an audit event "PAYMENT_FAILURE" should be recorded for "test_user" with the message containing "chapter1" and "chapter2"

Борьба с пропущенными условиями



* включая контекст, примеры и список установленных библиотек :)

Что можно исправить запуском?

- падение с missed mock statement
- исключения, из-за которых приложение не стартовало
- исключения, которые поймало пользовательское приложение
- неточные ассёрты

Что можно исправить запуском?



```
### Context ###
```

```
<...>
```

```
### Instructions ###
```

```
You will be given a test class with runtime errors.
```

```
Your task is to fix test setup and assertions to avoid failing tests.
```

```
<...>
```

```
### Test class ###
```

```
<...>
```

```
    .andExpect(status().isBadRequest());
```

```
<...>
```

```
<ERROR>
```

```
Error in method testPaymentFailureDueToInvalidStripeToken():  
java.lang.AssertionError: Status expected:<400> but was:<417>
```

```
    at org.springframework.test.util.AssertionErrors.fail(AssertionErrors.java:59)
```

```
    <...>
```

```
    at com.asanka.rest.PaymentControllerTest.testWithInvalidStripeToken(PaymentControllerTest.java:143)
```

```
</ERROR>
```

Если хорошие тесты так и не получились?




Совершенства нет, но...

Но есть:

- тестовые сценарии
- заготовка тестового класса
- возможность исправить один тест и регенерировать остальные


Formal methods


- LLM использует в тестах “случайные”, хоть и реалистичные данные
- Альтернатива – символьное исполнение и фаззинг


Смотреть запись 


ДОКЛАД

Символьное исполнение в .NET: Автоматическое тестирование, верификация и синтез программ

 **Дмитрий Иванов**
Huawei


 **Дмитрий Мордвинов**
JetBrains Research


RU 


Смотреть запись 

ДОКЛАД

Знакомство с фаззерами

 **Максим Пелевин**
Huawei


RU  Testing Tooling


Смотреть запись 

ДОКЛАД

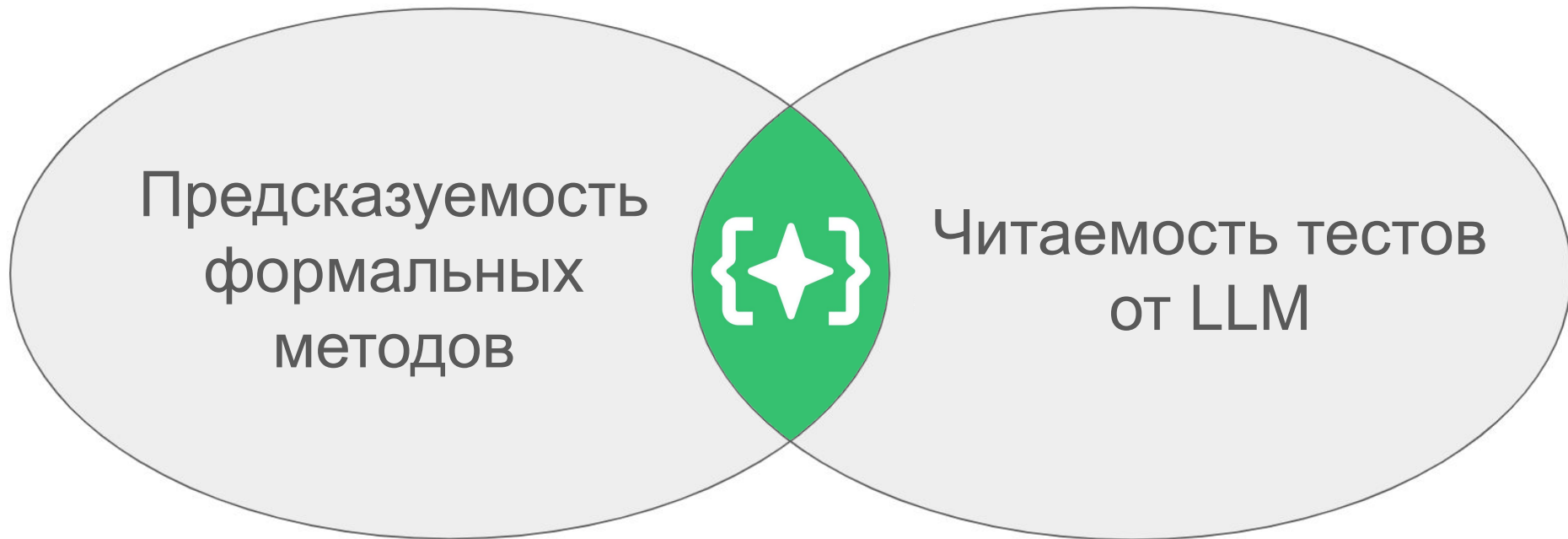
Анализ кода с нулевым числом ложных срабатываний в Java с помощью динамического символьного исполнения

 **Дмитрий Иванов**
Huawei

 **Дмитрий Мордвинов**

RU  Tooling

Formal Methods + Artificial Intelligence?



Резюмируем: как получить тест?

- Подготовить проект (например, установить Mosquito)
- Сформировать контекст для запроса (много аспектов)
- Дать дополнительные указания LLM
- Сгенерировать сценарии
- Сгенерировать по сценариям тесты
- Положить ответ в правильное место в проекте
- Добиться компилируемости (всеми доступными способами)
- Внести коррективы на основе запуска
- Потом опять добиться компилируемости :)



Резюмируем: как получить тест?

- Подготовить проект (например, установить Mockito)
- Сформировать контекст для запроса (много аспектов)
- Дать дополнительные указания LLM
- Сгенерировать сценарии
- Сгенерировать по сценариям тесты
- Положить ответ в правильное место в проекте
- Добиться компилируемости (всеми доступными способами)
- Внести коррективы на основе запуска
- Потом опять добиться компилируемости :)





- <https://explyt.com/>
- 40+ разработчиков
- 4 кандидата физико-математических наук
- ex. JetBrains Research, Huawei RRI
- выпускники СПбГУ, ИТМО, ВШЭ

Explyt Spring



Your teammate who makes development
productive

Plugin for Spring with Java and Kotlin support

Explyt Test



Your teammate who generates clear tests

Based on the latest AI research with
post-processing of world-leading static and dynamic
code analysis techniques



Егор Куликов
@egor_k_kulikov



Илья Муравьев
@ilyamuravjov