



**PKL**

Андрей Зонов  
Руководитель группы  
Stuff iOS





# Андрей Зонов

## Stuff iOS | Руководитель группы

 [linkedin.com/in/avzonov](https://www.linkedin.com/in/avzonov)

 [t.me/avzonov](https://t.me/avzonov)



# РКІ (Пикл)

- Язык программирования для связи статических ЯП
- Язык созданный для работы с конфигурациями
- Внимание на безопасности и поддержке
- Поддержка языков и IDE на старте

# Сегодня



Какую проблему решает PKL



Основные альтернативы



Примеры использования



Вывод

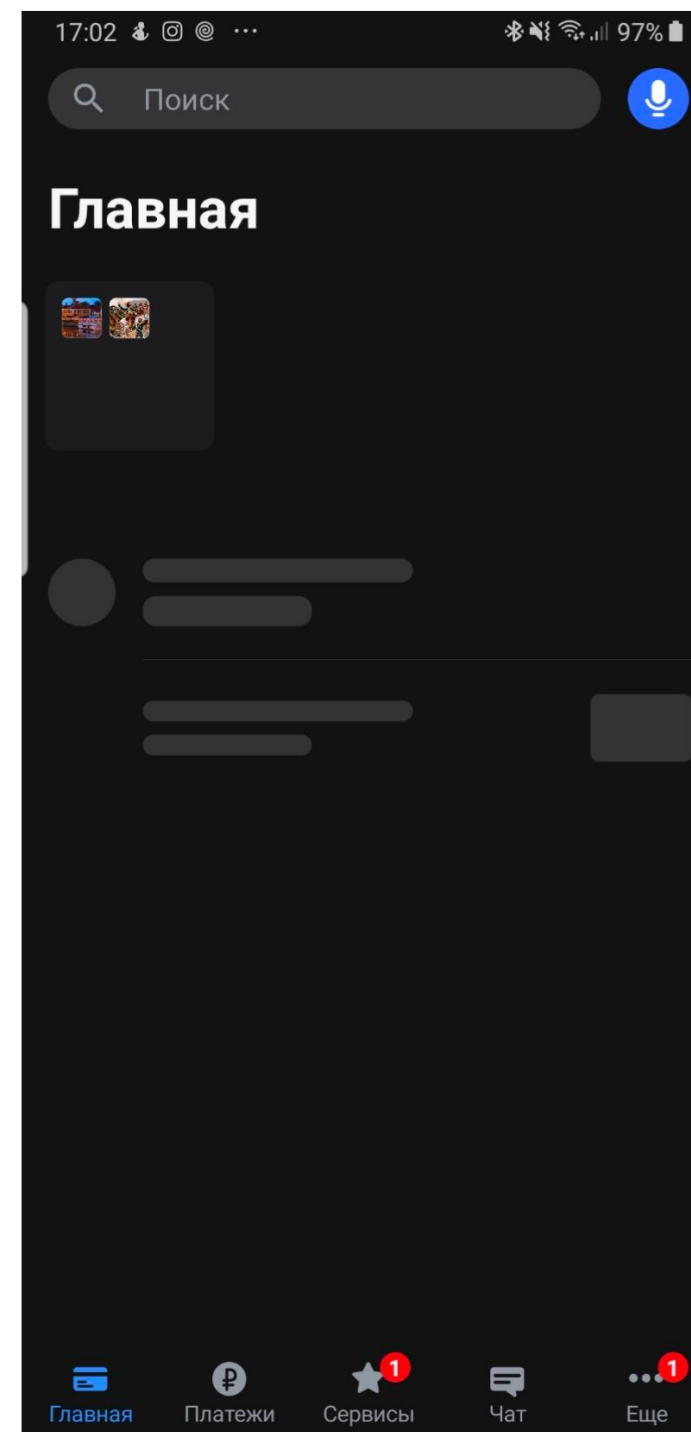
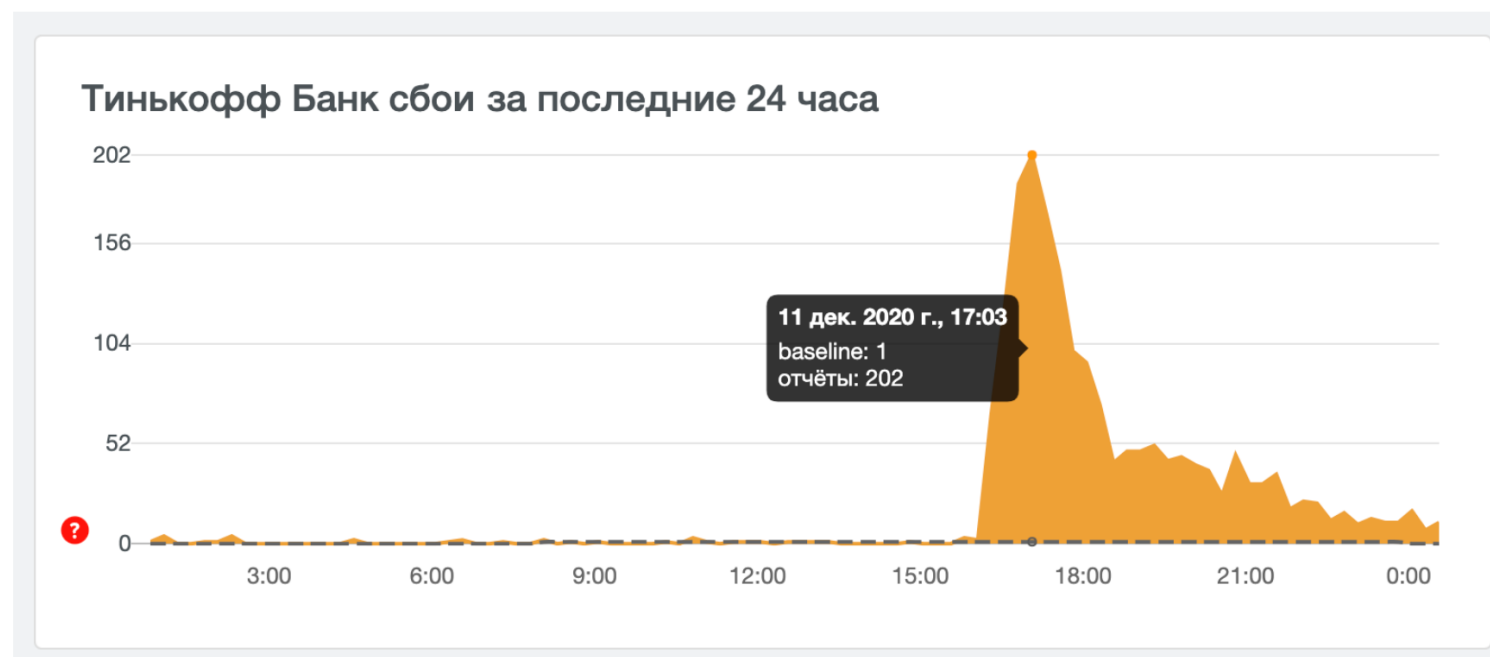
**Какую проблему решает РКИ?**



# Работа с JSON без тестов

# Зачем?

## Всё упало



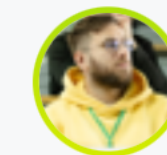
Смотреть запись



ДОКЛАД

Зал 1

«А у нас сейчас всё норм работает?» или Что такое observability мобильного приложения



Даниэль Халиулин  
Тинькофф

RU



Инфраструктура



# Зачем?

## Всё упало

API

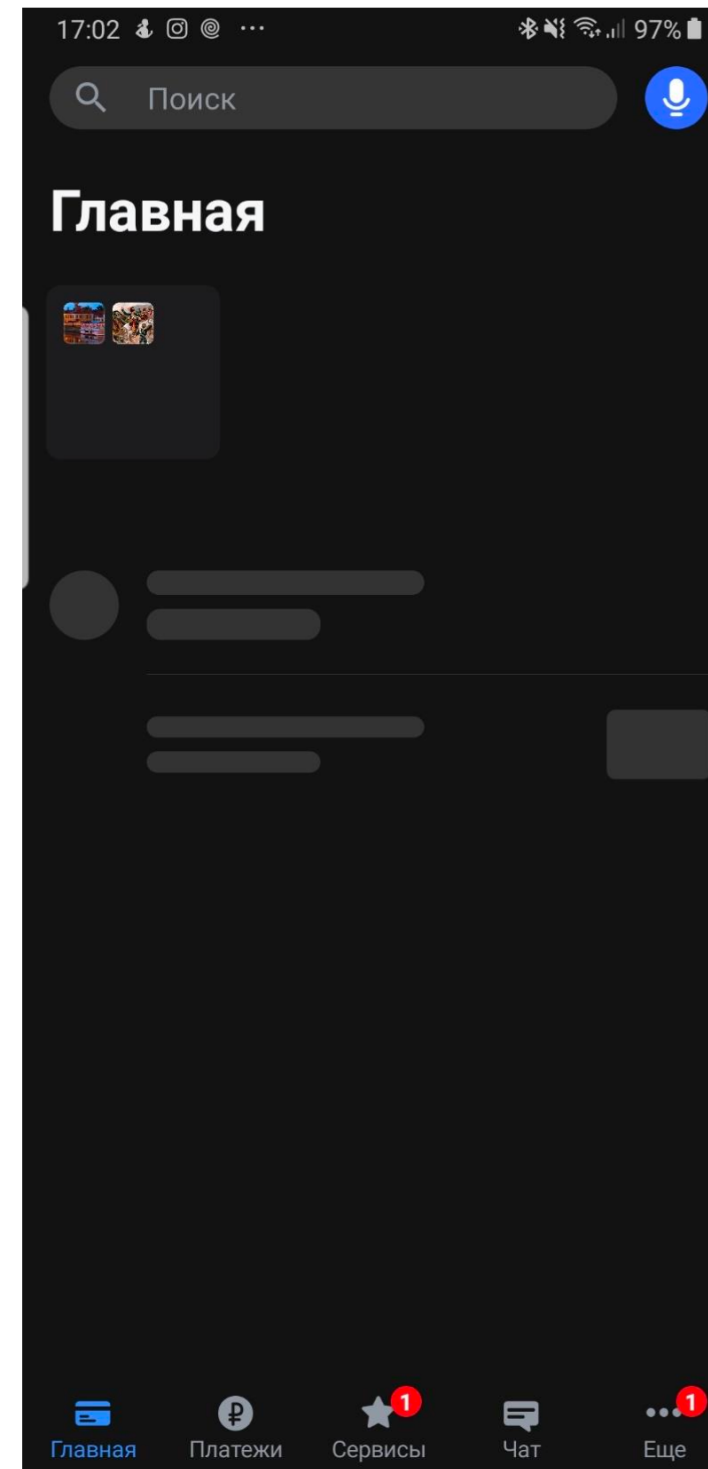
Кэши

Метрики

Бэкенды

Базы данных

Последние релизы сервисов



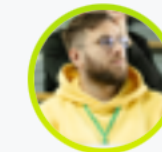
Смотреть запись



ДОКЛАД

Зал 1

«А у нас сейчас всё норм работает?» или Что такое observability мобильного приложения



Даниэль Халиулин  
Тинькофф

RU



Инфраструктура





# Зачем?

## Всё упало

API

Кэши

Метрики

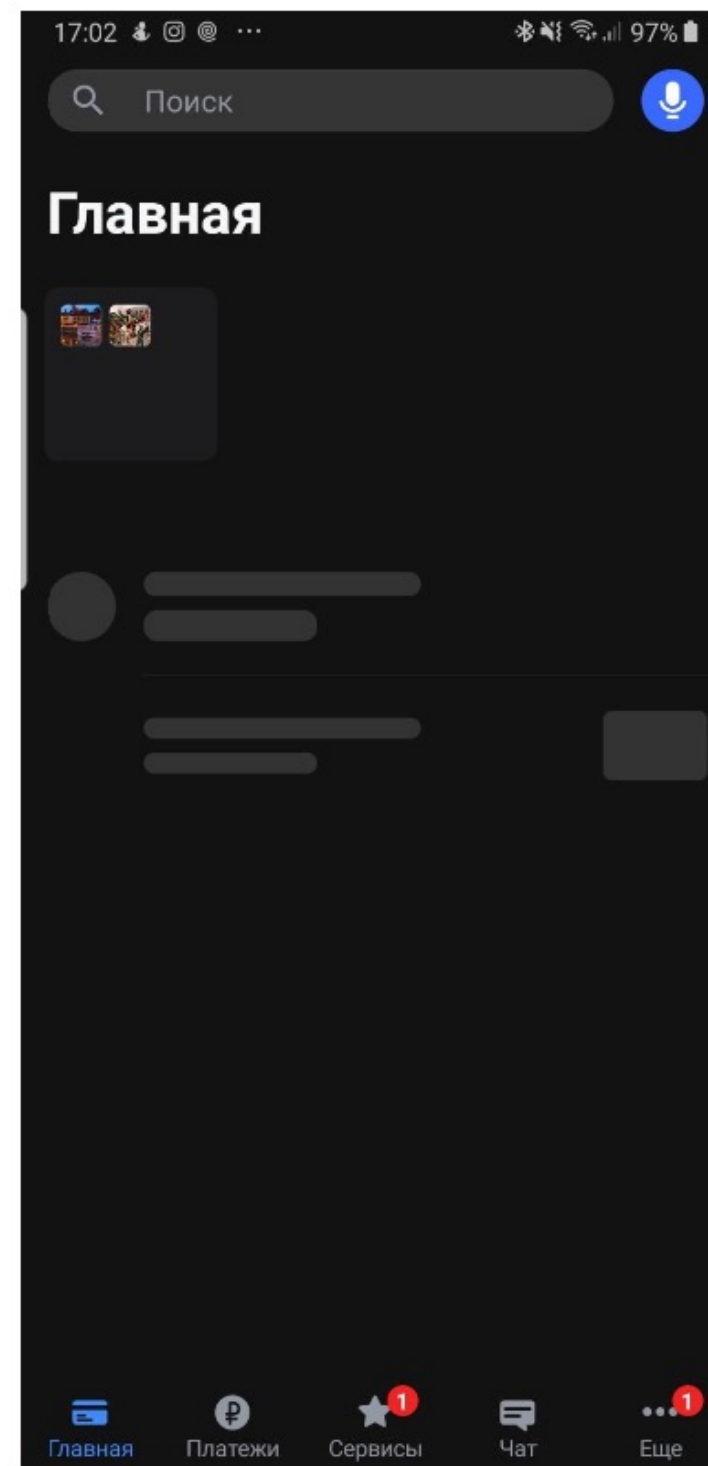
Бэкенды

Базы данных

Последние релизы сервисов



Изменения в конфигугах



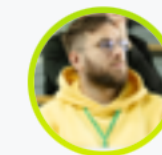
Смотреть запись



ДОКЛАД

Зал 1

«А у нас сейчас всё норм работает?» или Что такое observability мобильного приложения



Даниэль Халиулин  
Тинькофф

RU



Инфраструктура



# Зачем?

## Всё упало

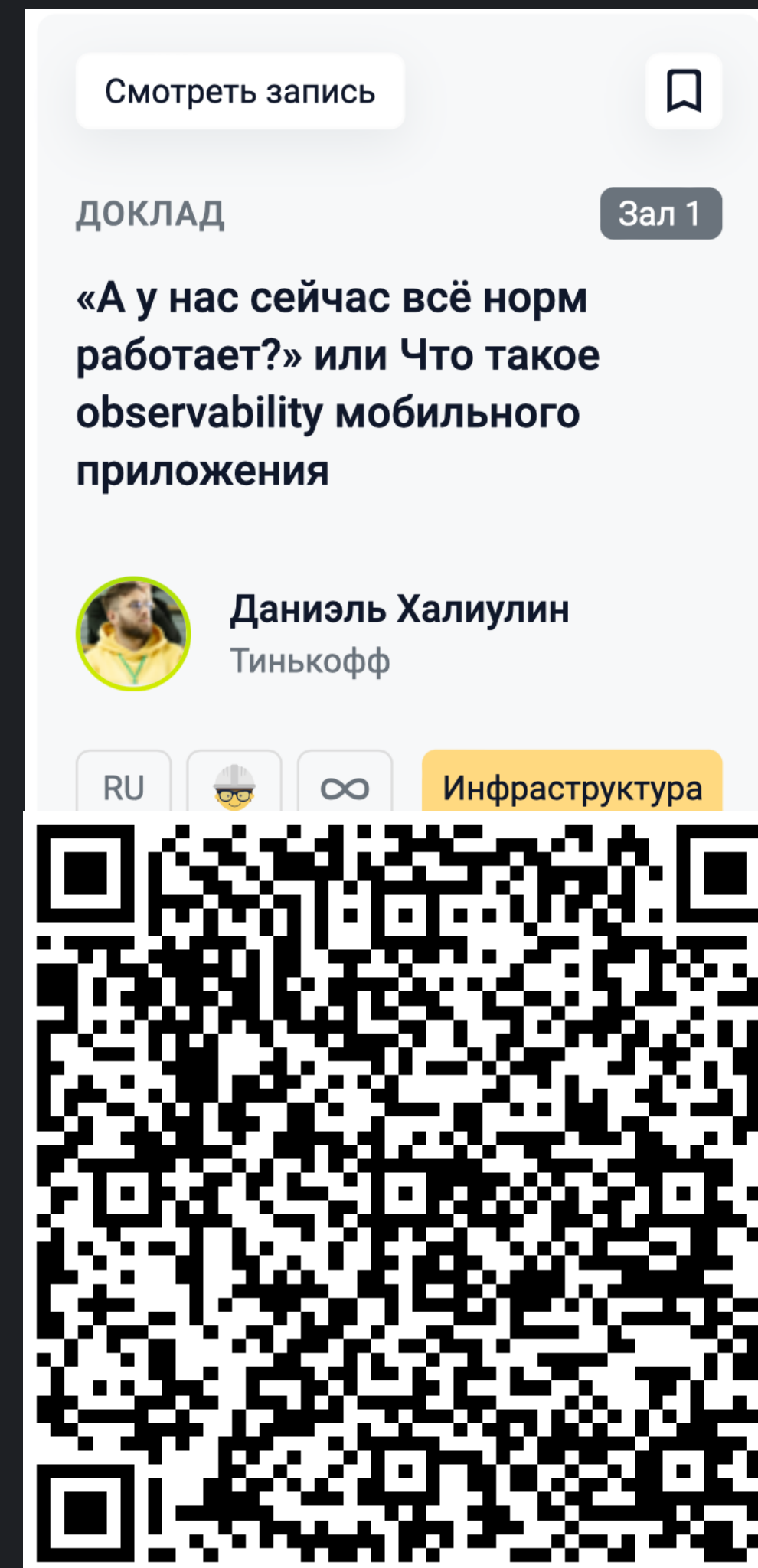
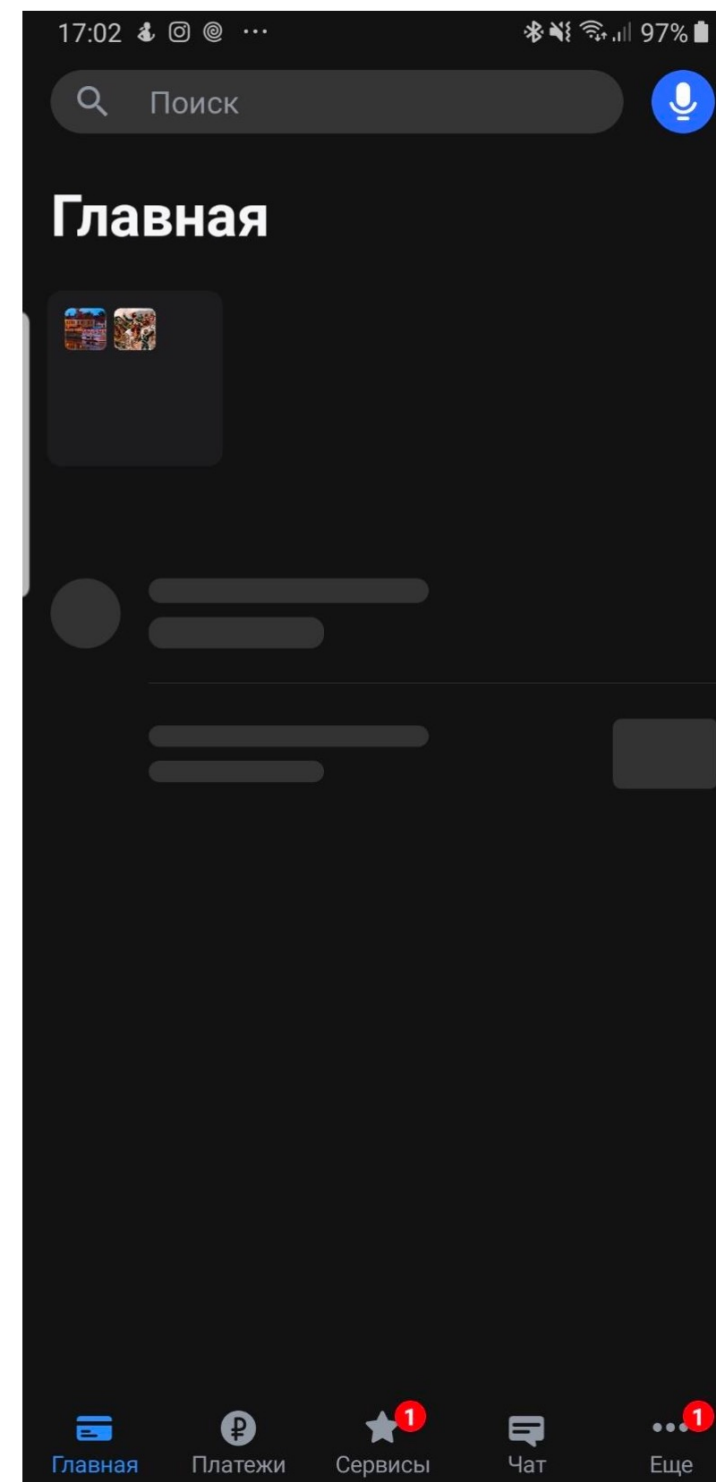
### Ожидание

```
{  
  "A": "aaaa",  
  "B": "bbbb",  
  "C": "cccc"  
}
```

← обязательное

### Реальность

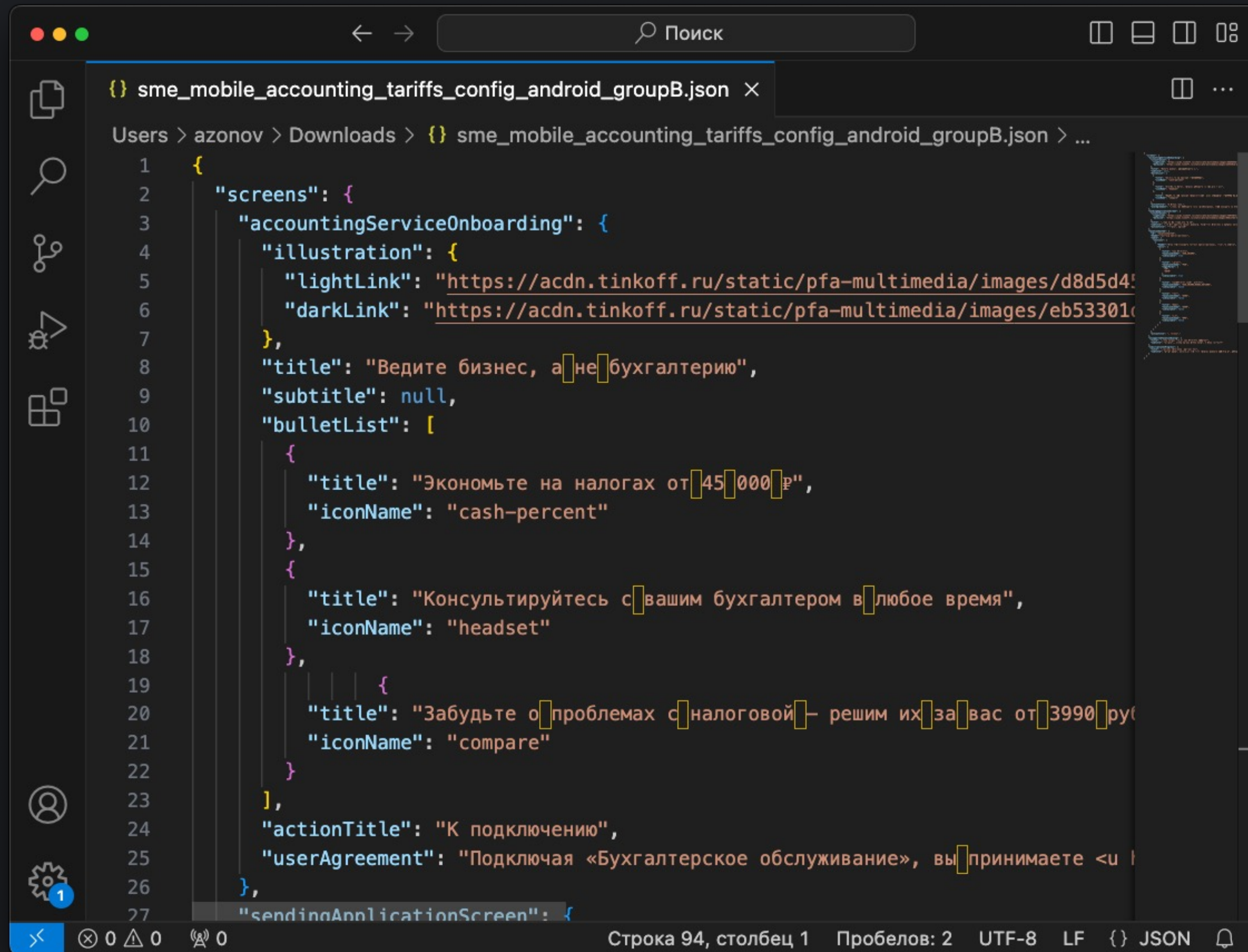
```
{  
  "A": 42,  
  "C": "cccc"  
  "D": [  
  ]  
}
```





# **Внедрение валидации контрактов**

# Как обезопасить прод?



```
1  {
2  "screens": {
3    "accountingServiceOnboarding": {
4      "illustration": {
5        "lightLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/d8d5d45...",
6        "darkLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/eb53301c...",
7      },
8      "title": "Ведите бизнес, а не бухгалтерию",
9      "subtitle": null,
10     "bulletList": [
11       {
12         "title": "Экономьте на налогах от 45 000 Р",
13         "iconName": "cash-percent"
14       },
15       {
16         "title": "Консультируйтесь с вашим бухгалтером в любое время",
17         "iconName": "headset"
18       },
19       {
20         "title": "Забудьте о проблемах с налоговой — решим их за вас от 3990 руб",
21         "iconName": "compare"
22       }
23     ],
24     "actionTitle": "К подключению",
25     "userAgreement": "Подключая «Бухгалтерское обслуживание», вы принимаете <u>и</u>
26   },
27   "sendingApplicationScreen": {
```

- Json

# Как обезопасить прод?

The screenshot shows an IDE with two code editors. The left editor displays Kotlin code for a data class named 'Screens'. The right editor displays Swift code for a struct named 'Screens' and another struct named 'TaxationSystem'. The Kotlin code includes imports for 'Keep' and 'Serializable', and a list of properties for the 'Screens' class. The Swift code defines the 'Screens' struct with properties for 'accountingServiceOnboarding', 'taxationSystem', 'unsupportedTaxationDialog', 'employees', 'howManyEmployees', 'sendingApplication', 'subscriptionInProgress', 'subscriptionCallDone', 'subscriptionPaid', 'usnIncome', 'onlineAccountingConnected', 'onlineAccountingDeclined', and 'tariffs'. It also defines the 'TaxationSystem' struct with properties for 'title', 'taxationScheme', and 'legalForms'. The status bar at the bottom indicates 'Строка 24, столбец 1 Пробелов: 4 UTF-8 LF Kotlin'.

```
package tariffkotlinspecdomainaccounting

import androidx.annotation.Keep
import java.io.Serializable

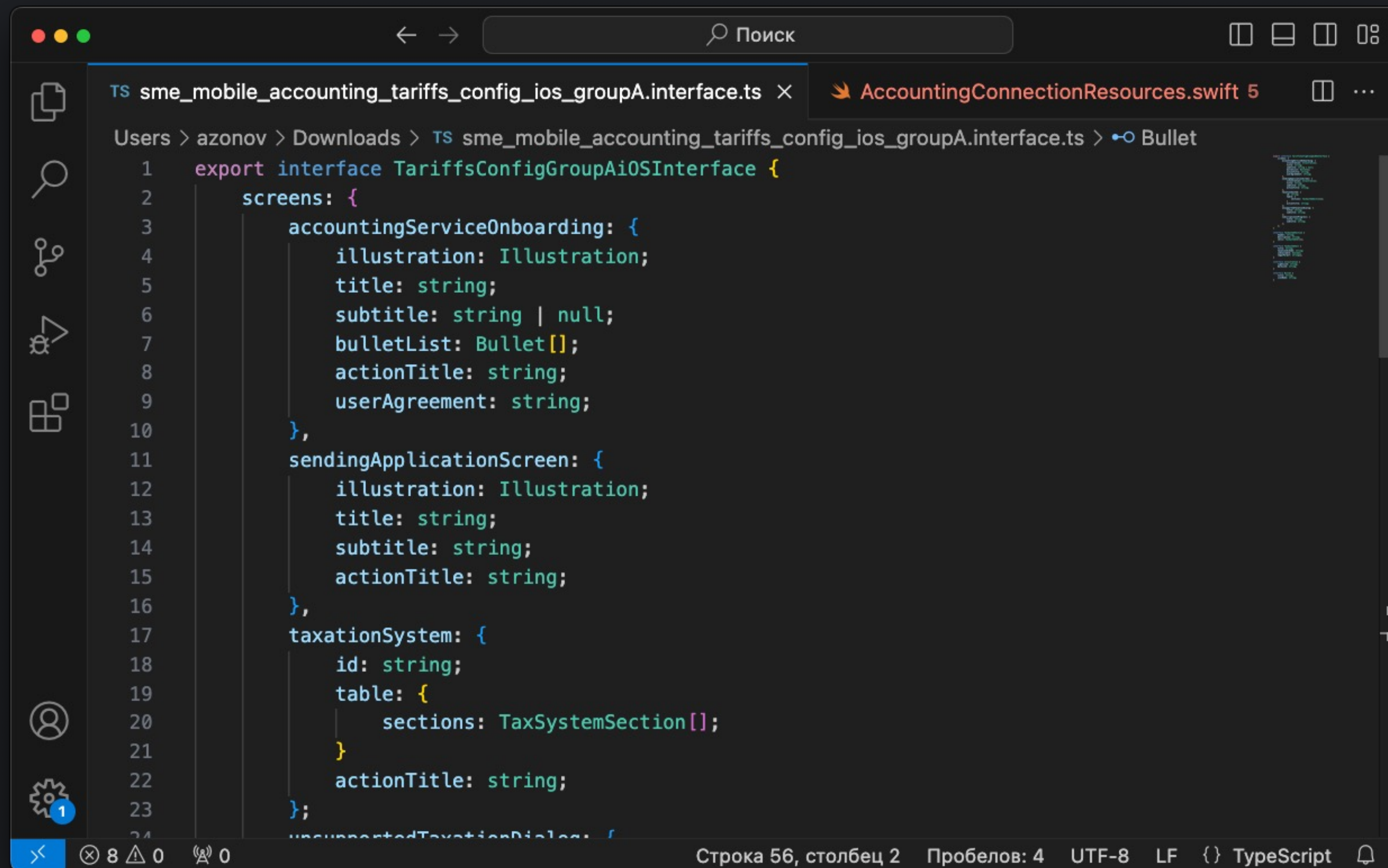
@Keep
data class Screens(
    val accountingServiceOnboarding: AccountingServiceOnboarding,
    val taxationSystem: TaxationSystem,
    val unsupportedTaxationDialog: UnsupportedTaxationDialog,
    val employees: Employees,
    val howManyEmployees: HowManyEmployees,
    val sendingApplication: SendingApplication,
    val subscriptionInProgress: AccountingSubscriptionInProgress,
    val subscriptionCallDone: AccountingSubscriptionCallDone,
    val subscriptionPaid: AccountingSubscriptionPaid,
    val usnIncome: UsnIncome,
    val onlineAccountingConnected: OnlineAccountingConnected,
    val onlineAccountingDeclined: OnlineAccountingDeclined,
    val tariffs: TariffsConfigData,
    val subscriptionCallDoneSelf: AccountingSubscriptionCallDoneSelf,
    val subscriptionPaidSelf: AccountingSubscriptionPaidSelf
) : Serializable
```

```
struct AccountingConnectionResources: Decodable {
    let accountingServiceOnboarding: AccountingServiceOnboarding
    let taxationSystem: TaxationSystem
    let unsupportedTaxationDialog: UnsupportedTaxationDialog
    let employees: Employees
    let howManyEmployees: HowManyEmployees
    let sendingApplication: SendingApplication
    let subscriptionInProgress: AccountingSubscriptionInProgress
    let subscriptionCallDone: AccountingSubscriptionCallDone
    let subscriptionPaid: AccountingSubscriptionPaid
    let usnIncome: UsnIncome
    let onlineAccountingConnected: OnlineAccountingConnected
    let onlineAccountingDeclined: OnlineAccountingDeclined
    let tariffs: Tariffs
    let subscriptionCallDoneSelf: AccountingSubscriptionCallDoneSelf
    let subscriptionPaidSelf: AccountingSubscriptionPaidSelf
}

struct TaxationSystem: Decodable, Hashable {
    let title: String
    let taxationScheme: TaxationScheme
    let legalForms: [LegalForm]?
```

- Json
- Swift
- Kotlin
- JS

# TypeScript?



The screenshot shows an IDE window with two tabs: 'TS sme\_mobile\_accounting\_tariffs\_config\_ios\_groupA.interface.ts' and 'AccountingConnectionResources.swift 5'. The active tab displays the following TypeScript code:

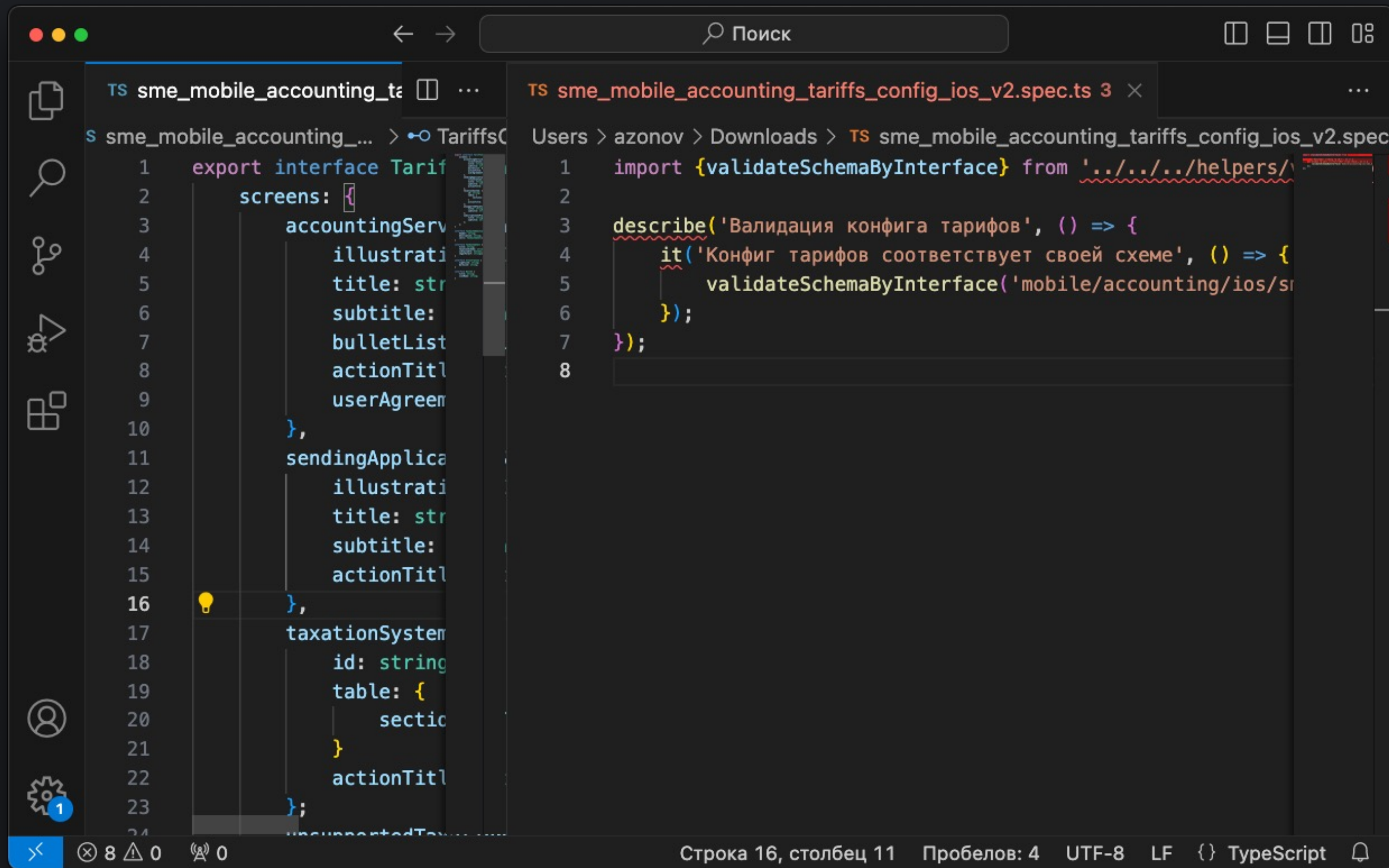
```
1 export interface TariffsConfigGroupAiOSInterface {
2   screens: {
3     accountingServiceOnboarding: {
4       illustration: Illustration;
5       title: string;
6       subtitle: string | null;
7       bulletList: Bullet[];
8       actionTitle: string;
9       userAgreement: string;
10    },
11    sendingApplicationScreen: {
12      illustration: Illustration;
13      title: string;
14      subtitle: string;
15      actionTitle: string;
16    },
17    taxationSystem: {
18      id: string;
19      table: {
20        sections: TaxSystemSection[];
21      }
22      actionTitle: string;
23    };
24  };
25 }
```

The status bar at the bottom indicates: 'Строка 56, столбец 2 Пробелов: 4 UTF-8 LF {} TypeScript'.

- Json
- Swift
- Kotlin
- JS
- TypeScript



# TypeScript?



```
TS sme_mobile_accounting_tariffs_config_ios_v2.spec.ts 3 x
Users > azonov > Downloads > TS sme_mobile_accounting_tariffs_config_ios_v2.spec.ts
1 import {validateSchemaByInterface} from '../helpers/validators';
2
3 describe('Валидация конфига тарифов', () => {
4   it('Конфиг тарифов соответствует своей схеме', () => {
5     validateSchemaByInterface('mobile/accounting/ios/specs/tariffs-config', TariffsConfig);
6   });
7 });
```

```
TS sme_mobile_accounting_tariffs_config_ios_v2.spec.ts
1 export interface TariffsConfig {
2   screens: {
3     accountingService: AccountingService;
4     illustration: Illustration;
5     title: string;
6     subtitle: string;
7     bulletList: string;
8     actionTitle: string;
9     userAgreement: string;
10  },
11  sendingApplication: {
12    illustration: Illustration;
13    title: string;
14    subtitle: string;
15    actionTitle: string;
16  },
17  taxationSystem: {
18    id: string;
19    table: {
20      section: string;
21    };
22    actionTitle: string;
23  };
24  unsupportedTaxSystem: string;
25 }
```

- Json
- Swift
- Kotlin
- JS
- TypeScript



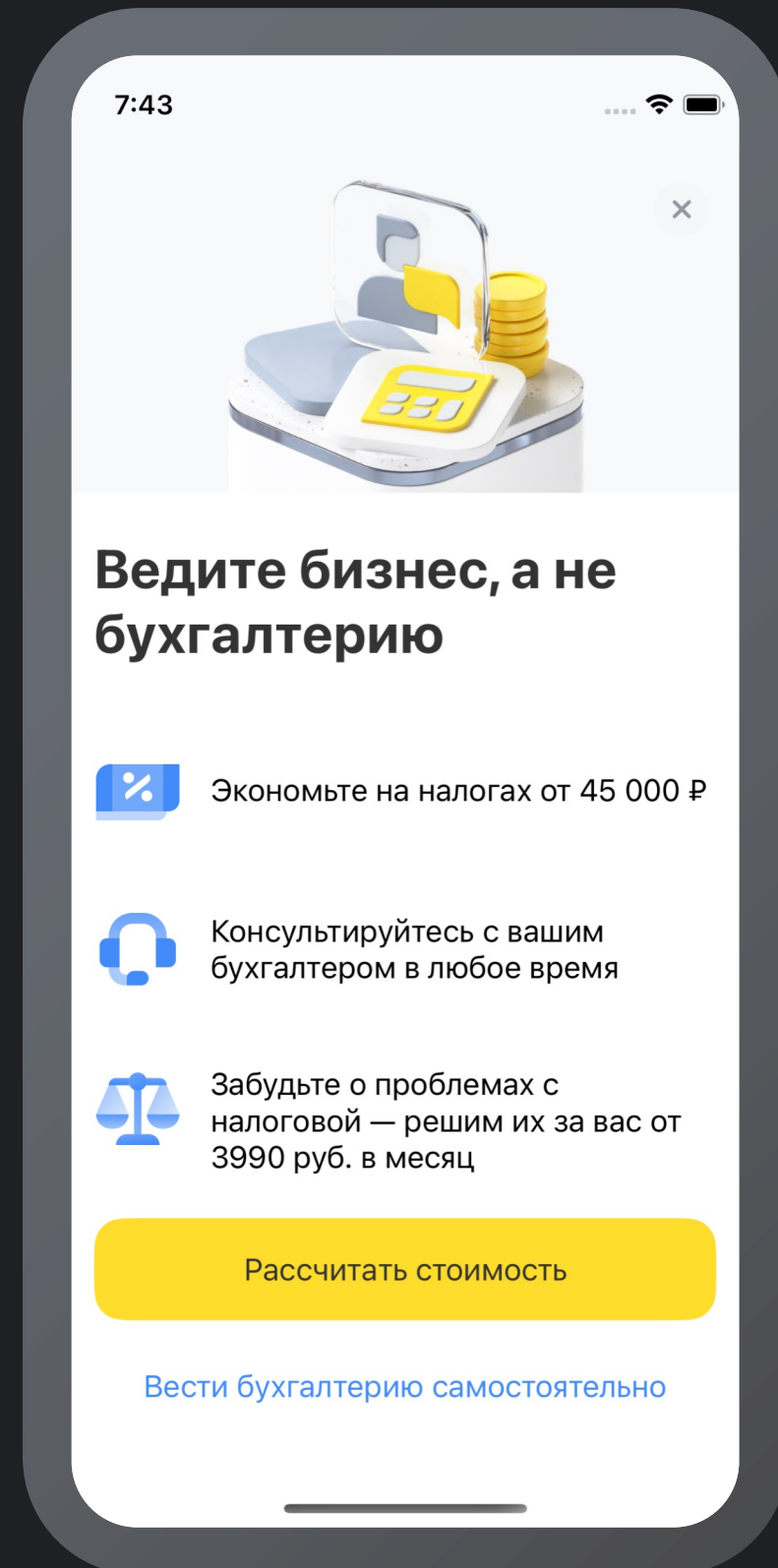
# Вы представляете что видит пользователь?

```
{} sme_mobile_accounting_tariffs_config_android_groupB.json x
Users > azonov > Downloads > {} sme_mobile_accounting_tariffs_config_android_groupB.json > ...
1  {
2    "screens": {
3      "accountingServiceOnboarding": {
4        "illustration": {
5          "lightLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/d8d5d45...",
6          "darkLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/eb53301c...",
7        },
8        "title": "Ведите бизнес, а не бухгалтерию",
9        "subtitle": null,
10       "bulletList": [
11         {
12           "title": "Экономьте на налогах от 45 000 Р",
13           "iconName": "cash-percent"
14         },
15         {
16           "title": "Консультируйтесь с вашим бухгалтером в любое время",
17           "iconName": "headset"
18         },
19         {
20           "title": "Забудьте о проблемах с налоговой - решим их за вас от 3990 руб",
21           "iconName": "compare"
22         }
23       ],
24       "actionTitle": "К подключению",
25       "userAgreement": "Подключая «Бухгалтерское обслуживание», вы принимаете <u>и</u>
26     },
27     "sendingApplicationScreen": {
```

Строка 94, столбец 1 Пробелов: 2 UTF-8 LF {} JSON



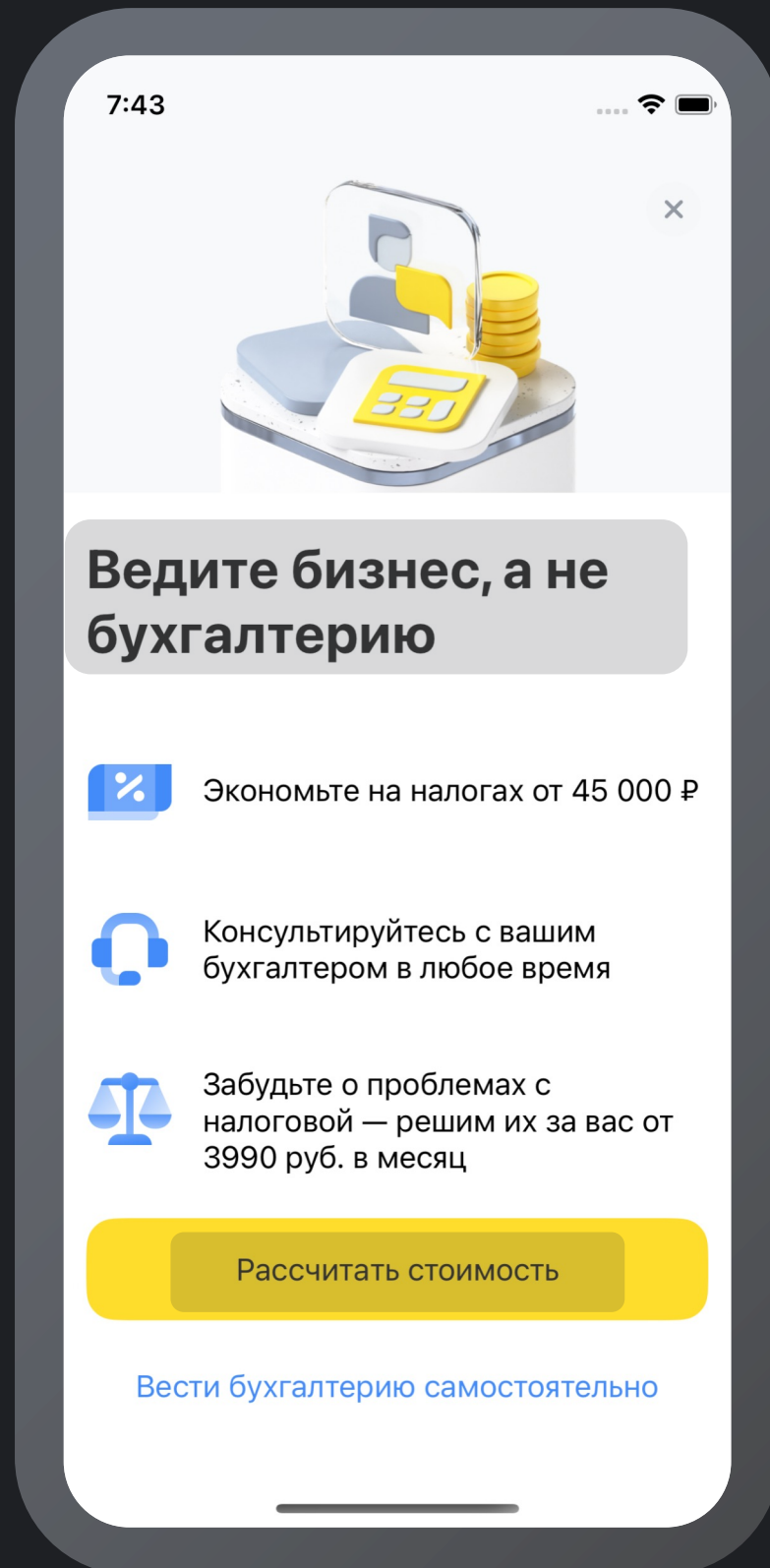
# Вы представляете что видит пользователь?



```
{} sme_mobile_accounting_tariffs_config_android_groupB.json ×
Users > azonov > Downloads > {} sme_mobile_accounting_tariffs_config_android_groupB.json > ...
1  {
2    "screens": {
3      "accountingServiceOnboarding": {
4        "illustration": {
5          "lightLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/d8d5d45...",
6          "darkLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/eb53301c...",
7        },
8        "title": "Ведите бизнес, а не бухгалтерию",
9        "subtitle": null,
10       "bulletList": [
11         {
12           "title": "Экономьте на налогах от 45 000 ₽",
13           "iconName": "cash-percent"
14         },
15         {
16           "title": "Консультируйтесь с вашим бухгалтером в любое время",
17           "iconName": "headset"
18         },
19         {
20           "title": "Забудьте о проблемах с налоговой — решим их за вас от 3990 руб.",
21           "iconName": "compare"
22         }
23       ],
24       "actionTitle": "К подключению",
25       "userAgreement": "Подключая «Бухгалтерское обслуживание», вы принимаете <u>
26     },
27     "sendingApplicationScreen": {
```

Строка 94, столбец 1 Пробелов: 2 UTF-8 LF {} JSON

# Ограничения на поля



```
{} sme_mobile_accounting_tariffs_config_android_groupB.json x
Users > azonov > Downloads > {} sme_mobile_accounting_tariffs_config_android_groupB.json > ...
1  {
2    "screens": {
3      "accountingServiceOnboarding": {
4        "illustration": {
5          "lightLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/d8d5d45...",
6          "darkLink": "https://acdn.tinkoff.ru/static/pfa-multimedia/images/eb53301c...",
7        },
8        "title": "Ведите бизнес, а не бухгалтерию",
9        "subtitle": null,
10       "bulletList": [
11         {
12           "title": "Экономьте на налогах от 45 000 ₽",
13           "iconName": "cash-percent"
14         },
15         {
16           "title": "Консультируйтесь с вашим бухгалтером в любое время",
17           "iconName": "headset"
18         },
19         {
20           "title": "Забудьте о проблемах с налоговой — решим их за вас от 3990 руб.",
21           "iconName": "compare"
22         }
23       ],
24       "actionTitle": "К подключению",
25       "userAgreement": "Подключая «Бухгалтерское обслуживание», вы принимаете <u>
26     },
27     "sendingApplicationScreen": {
```

Строка 94, столбец 1 Пробелов: 2 UTF-8 LF {} JSON

# Внедрение валидации контрактов



Контракты



Тесты



Вероятность ошибки парсинга



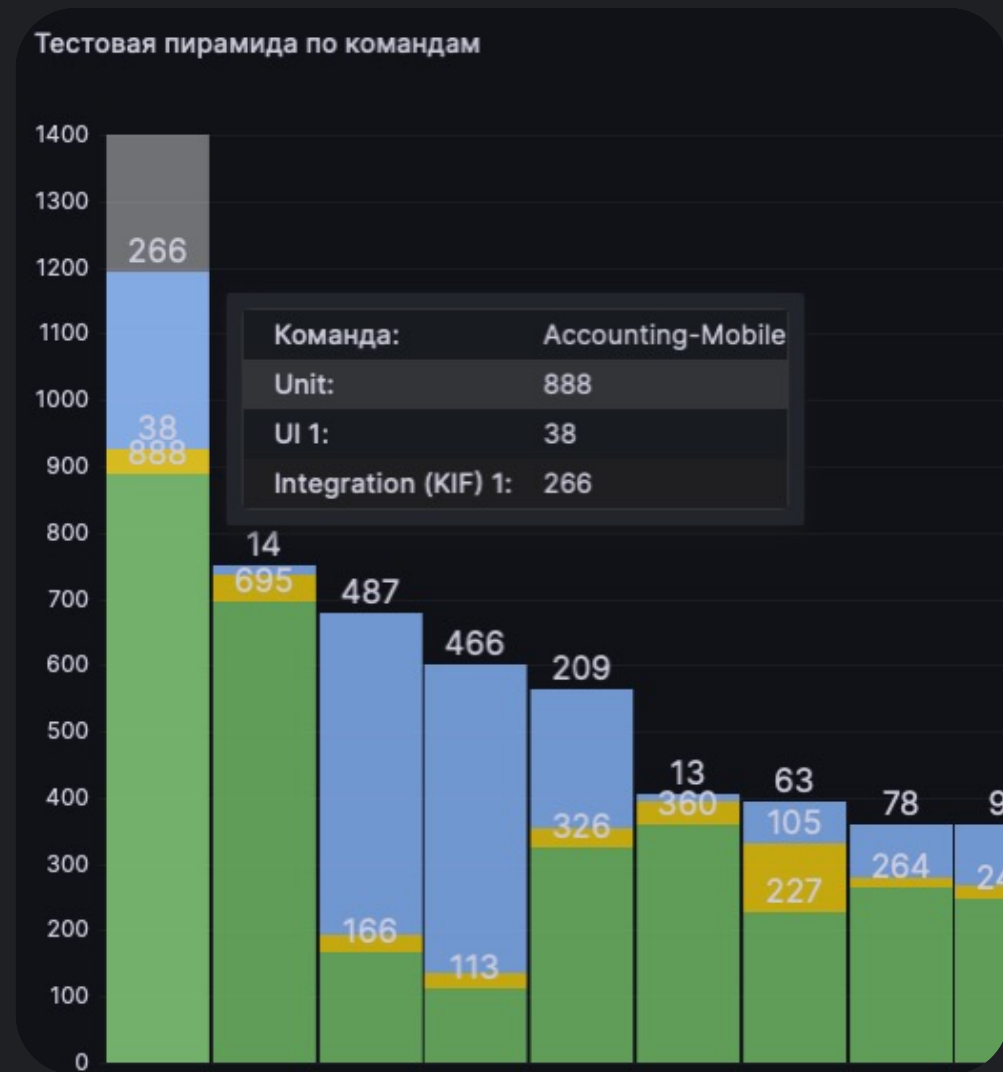
Только для конфигов



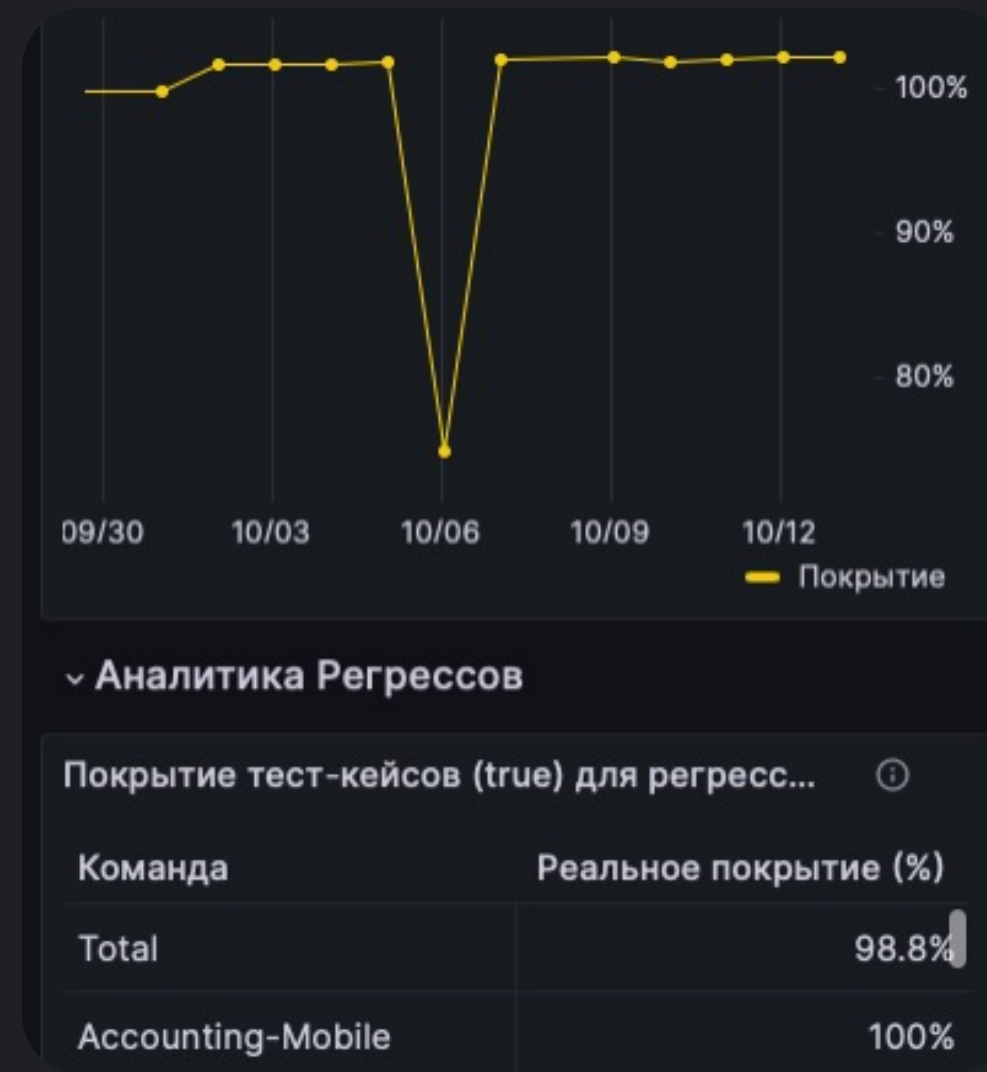
Сложно понять что меняешь



**Внедряем кодогенерацию**



Тестовая пирамида

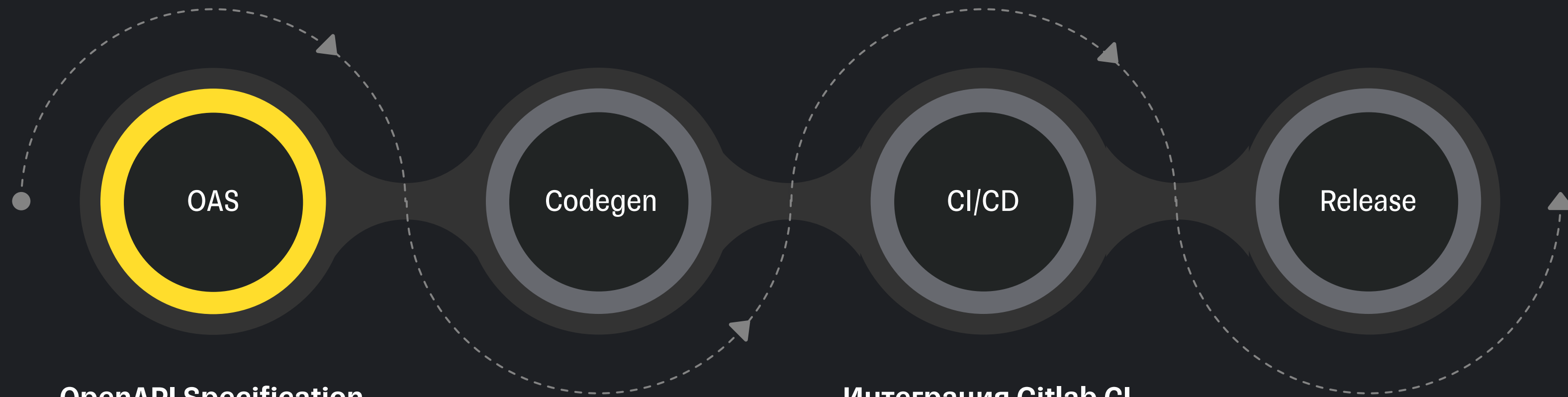


Покрытие



Стабильность

# Разработка через кодогенерацию



## OpenAPI Specification

Как единственный источник правды для клиента и сервера

## API сразу после спеки

Генерация замочанного API на сервере по спекам для параллельной разработки клиента и сервера

## Интеграция Gitlab CI

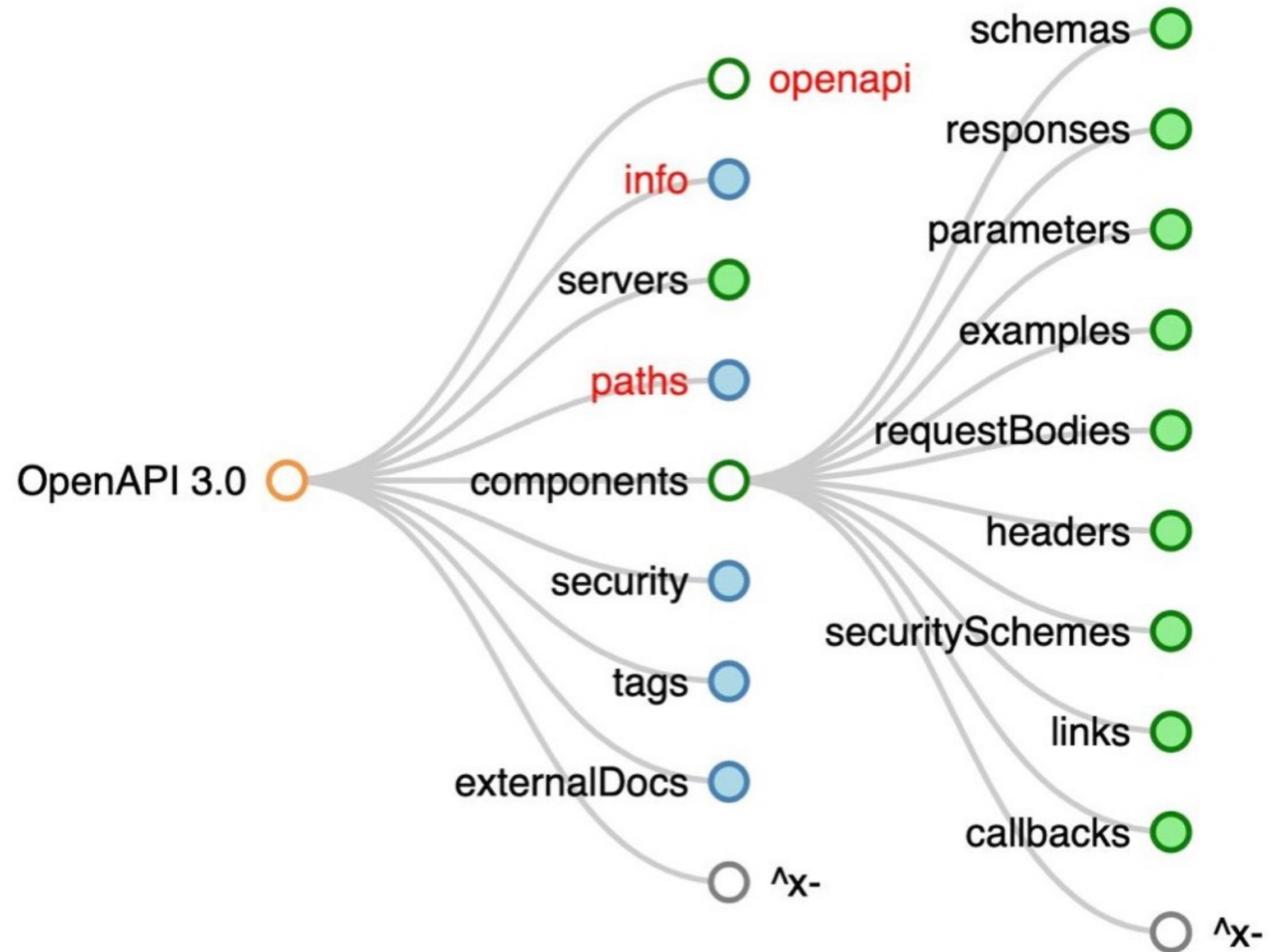
Pipeline который при изменении файла спецификации API будет регенерировать файлы для сервера и клиентов и говорить есть ли проблемы с совместимостью

## TBD

Контрактное тестирование из коробки доступно как потребителям так и создателям API, спим спокойно

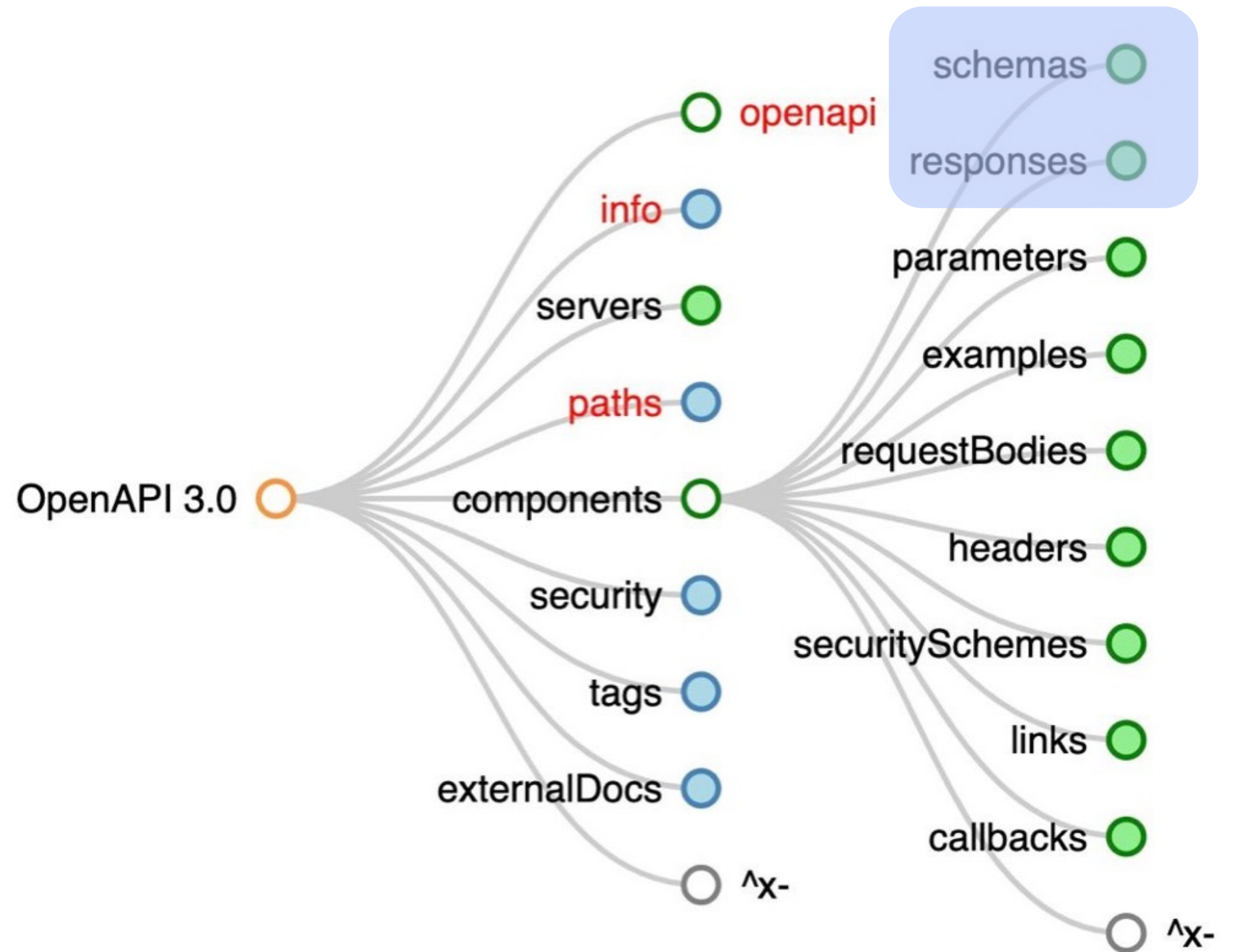
# OpenAPI

Документ YAML OpenAPI  
представляет собой дерево с  
набором ветвей



# OpenAPI

Документ YAML OpenAPI  
представляет собой дерево с  
набором ветвей





# OpenAPI

```
Category:  
  type: object  
  properties:  
    id:  
      type: integer  
      format: int64  
      example: 1  
    name:  
      type: string  
      example: Dogs  
  xml:  
    name: category
```

## Типы

string, number, integer,  
boolean, object, array

```
Category:  
  type: object  
  properties:  
    id:  
      type: integer  
      format: int64  
      example: 1  
    name:  
      type: string  
      example: Dogs  
  xml:  
    name: category
```

## Типы

string, number, integer,  
boolean, object, array

```
/users/{userId}:  
  get:  
    parameters:  
      - name: userId  
        in: path  
        required: true  
        description: "ID o  
        allowEmptyValue: t  
        schema:  
          type: integer
```

## Возможность задания ограничений

max, min, pattern, format, length,  
nullable, enum, required

```
Category:
  type: object
  properties:
    id:
      type: integer
      format: int64
      example: 1
    name:
      type: string
      example: Dogs
  xml:
    name: category
```

## Типы

string, number, integer,  
boolean, object, array

```
/users/{userId}:
  get:
    parameters:
      - name: userId
        in: path
        required: true
        description: "ID of user"
        allowEmptyValue: true
        schema:
          type: integer
```

## Возможность задания ограничений

max, min, pattern, format, length,  
nullable, enum, required

```
12 Available Clients: [ akka-scala,
11   android, async-scala, clojure, cpprest, c
10   cwiki, dart, dynamic-html, flash, go, groov
9   html2, java, javascript, javascript-clo
8   jaxrs-cxf-client, jmeter, objc, perl, php,
7   qt5cpp, ruby, scala, swagger, swagger-yaml
6   swift3, tizen, typescript-angular, typesc
5   typescript-fetch, typescript-node],
4
3 Available Servers: [ aspnet5, aspnetcore,
2   erlang-server, go-server, haskell, inflec
1   jaxrs, jaxrs-cxf, jaxrs-cxf-cdi, jaxrs-re
3   "jaxrs-spec", "lumen", "msf4j", "nancyfx"
1   python-flask, rails5, scalatra, silex-PHF
2   slim, spring, undertow]
```

## Кодогенерация

Jar исполняемый cli с кастомизацией  
через mustache

# OpenAPI

```
openapi: '3.1.0'
info:
  title: GreetingService
  version: 1.0.0
```

- Версия API и название

# OpenAPI

```
openapi.yaml No Preview ↙  
1 openapi: '3.1.0'  
2 info:  
3   title: GreetingService  
4   version: 1.0.0  
5 servers:  
6   - url: https://example.com/api  
7     description: Example service deployment.
```

- Версия API и название
- Список URL сервера

# OpenAPI

openapi.yaml

No Preview ↗

```
1 openapi: '3.1.0'
2 info:
3   title: GreetingService
4   version: 1.0.0
5 servers:
6   - url: https://example.com/api
7     description: Example service deployment.
8 paths:
9   /greet:
10    get:
11      operationId: getGreeting
```

- Версия API и название
- Список URL сервера
- Список путей

# OpenAPI

```
openapi.yaml No Preview ✓
1 openapi: '3.1.0'
2 info:
3   title: GreetingService
4   version: 1.0.0
5 servers:
6   - url: https://example.com/api
7     description: Example service deployment.
8 paths:
9   /greet:
10    get:
11      operationId: getGreeting
12      parameters:
13        - name: name
14          required: false
15          in: query
16          description: The name used in the returned greeting.
17          schema:
18            type: string
```

- Версия API и название
- Список URL сервера
- Список путей
- Параметры запроса

# OpenAPI

```
openapi.yaml No Preview ✓
1 openapi: '3.1.0'
2 info:
3   title: GreetingService
4   version: 1.0.0
5 servers:
6   - url: https://example.com/api
7     description: Example service deployment.
8 paths:
9   /greet:
10    get:
11      operationId: getGreeting
12      parameters:
13        - name: name
14          required: false
15          in: query
16          description: The name used in the returned greeting.
17          schema:
18            type: string
19    responses:
20      '200':
21        description: A success response with a greeting.
```

- Версия API и название
- Список URL сервера
- Список путей
- Параметры запроса
- Коды ответов



# OpenAPI

```
openapi.yaml No Preview ↗
1 openapi: '3.1.0'
2 info:
3   title: GreetingService
4   version: 1.0.0
5 servers:
6   - url: https://example.com/api
7     description: Example service deployment.
8 paths:
9   /greet:
10    get:
11      operationId: getGreeting
12      parameters:
13        - name: name
14          required: false
15          in: query
16          description: The name used in the returned greeting.
17          schema:
18            type: string
19      responses:
20        '200':
21          description: A success response with a greeting.
22          content:
23            application/json:
24              schema:
25                $ref: '#/components/schemas/Greeting'
26 components:
27   schemas:
28     Greeting:
29       type: object
30       properties:
31         message:
32           type: string
33       required:
34         - message
```

- Версия API и название
- Список URL сервера
- Список путей
- Параметры запроса
- Коды ответов
- Тело ответа

# OpenAPI

```
openapi.yaml No Preview ↗
1 openapi: '3.1.0'
2 info:
3   title: GreetingService
4   version: 1.0.0
5 servers:
6   - url: https://example.com/api
7     description: Example service deployment.
8 paths:
9   /greet:
10    get:
11      operationId: getGreeting
12      parameters:
13        - name: name
14          required: false
15          in: query
16          description: The name used in the returned greeting.
17          schema:
18            type: string
19      responses:
20        '200':
21          description: A success response with a greeting.
22          content:
23            application/json:
24              schema:
25                $ref: '#/components/schemas/Greeting'
26 components:
27   schemas:
28     Greeting:
29       type: object
30       properties:
31         message:
32           type: string
33       required:
34         - message
```

- Версия API и название
- Список URL сервера
- Список путей
- Параметры запроса
- Коды ответов
- Тело ответа

# OpenAPI

```
8 - paths:
9 -   /greet:
10 -     get:
11 -       operationId: getGreeting
12 -       parameters:
13 -         - name: name
14 -           required: false
15 -           in: query
16 -           description: The name used in the returned greeting.
17 -           schema:
18 -             type: string
19 -       responses:
20 -         '200':
21 -           description: A success response with a greeting.
22 -           content:
23 -             application/json:
24 -               schema:
25 -                 $ref: '#/components/schemas/Greeting'
26 - components:
27 -   schemas:
28 -     Greeting:
29 -       type: object
30 -       properties:
31 -         message:
32 -           type: string
33 -       required:
34 -         - message
```

Servers

https://example.com/api - Example service deployment. ▾

default ^

GET /greet ^

Parameters

Try it out

Name	Description
name	The name used in the returned greeting.
string	<input type="text" value="name"/>
(query)	

Responses

Code	Description	Links
200	A success response with a greeting.	No links

Media type

application/json ▾

Controls Accept header.

Example Value | Schema

```
{
  "message": "string"
}
```

# Codegen

OpenAPI

main.swift

```
import Foundation
import Vapor
import OpenAPIRuntime
import OpenAPIVapor

// Define a type that conforms to the generated protocol.
struct GreetingServiceAPIImpl: APIProtocol {
    func getGreeting(
        _ input: Operations.getGreeting.Input
    ) async throws -> Operations.getGreeting.Output {
        let name = input.query.name ?? "Stranger"
        let greeting = Components.Schemas.Greeting(message: "Hello, \(name)")
        return .ok(.init(body: .json(greeting)))
    }
}
```

Сервер

# Codegen

OpenAPI

main.swift

```
import Foundation
import Vapor
import OpenAPIRuntime
import OpenAPIVapor

// Define a type that conforms to the generated protocol.
struct GreetingServiceAPIImpl: APIProtocol {
    func getGreeting(
        _ input: Operations.getGreeting.Input
    ) async throws -> Operations.getGreeting.Output {
        let name = input.query.name ?? "Stranger"
        let greeting = Components.Schemas.Greeting(message: "Hello, \(name)")
        return .ok(.init(body: .json(greeting)))
    }
}
```

Сервер

main.swift

```
import OpenAPIRuntime
import OpenAPIURLSession

let client = Client(
    serverURL: try Servers.server2(),
    transport: URLSessionTransport()
)

let response = try await client.getGreeting(query: .init(name: "CLI"))
switch response {
case .ok(let okResponse):
    switch okResponse.body {
    case .json(let greeting):
        print(greeting.message)
    }
case .undocumented(statusCode: let statusCode, _):
    print("😞 undocumented response: \(statusCode)")
}
```

Клиент

# OpenAPI Codegen

```
resources

PROВОДНИК
RESOU...
  > AccountingWebCod...
  > SmeAndroidCodege...
  > SmelosCodegenGen...
  > SmeKotlinBackendC...
  > SmeKotlinBackendS...
  > СТРУКТУРА
  > ВРЕМЕННАЯ ШКАЛА

api.mustache x
SmelosCodegenGenerator > api.mustache
26 }{{/isEnum}}{{/parameters}}{{/contents}}{{/operation}}
27 }{{/operations}}
28
29 class {{#titlecase}}{{projectName}}{{/titlecase}}Ser
30
31 // MARK: Private Properties
32
33 private let environment: {{projectName}}Environm
34 private let networkService: NetworkServiceProtoc
35
36 // MARK: Lifecycle
37
38 init(environment: {{projectName}}EnvironmentConf
39     self.environment = environment
40     self.networkService = networkService
41 }
42
43 // MARK: Public
44 {{#operations}}{{#operation}}{{#contents}}
45 func {{operationId}}({{#parameters}}{{paramName}}
46     let parameters = {{#titlecase}}{{operationId}}
47     let parser = {{#returnType}}CordableParser<{{
48     let target = {{#titlecase}}{{operationId}}{{
49     return networkService
50         .sendRequest(target: target, parser: par
51         .eraseToAnyPublisher()
52     }
53 }{{/contents}}{{/operation}}
54 }{{/operations}}
55 }

api.mustache x
SmeAndroidCodegenGenerator > api.mustache
10 {{#imports}}import {{import}};
11 }{{/imports}}
12
13 {{#operations}}
14 interface {{classname}} {
15     {{#operation}}
16     /**
17     * {{summary}}
18     * {{notes}}
19     {{#allParams}}
20     * @param {{paramName}} {{description}}{{#req
21     }{{/allParams}}
22     * @return {{#returnType}}{{returnType}}{{/retu
23     }{{#isDeprecated}}
24     * @deprecated
25     }{{/isDeprecated}}
26     {{#externalDocs}}
27     * {{description}}
28     * @see <a href="{{url}}">{{summary}} Document
29     }{{/externalDocs}}
30     */
31     {{#isDeprecated}}
32     @Deprecated
33     }{{/isDeprecated}}
34     {{#formParams}}
35     {{#@first}}
36     {{#isMultipart}}@retrofit2.http.Multipar
37     }{{/@first}}
38     }{{/formParams}}
```

Строка 1, столбец 1 Пробелов: 4 UTF-8 LF Mustache (HTML)

# OpenAPI Codegen

```
SmelosCodegenGenerator > api.mustache
26 {{{/isnum}}}{{/parameters}}}{{/contents}}}{{/operation
27   }{/operations}}
28
29 class {{#titlecase}}>{{projectName}}{/{titlecase}}Ser
30
31   // MARK: Private Properties
32
33   private let environment: {{projectName}}Environm
34   private let networkService: NetworkServiceProtoc
35
36   // MARK: Lifecycle
37
38   init(environment: {{projectName}}EnvironmentConf
39     self.environment = environment
40     self.networkService = networkService
41   }
42
43   // MARK: Public
44   {{{#operations}}}{#{operation}}}{#{contents}}
45   func {{operationId}}({{{#parameters}}}{paramName}
46     let parameters = {{{#titlecase}}}{operationId
47     let parser = {{{#returnType}}}{CodableParser}<{{
48     let target = {{{#titlecase}}}{operationId}}{
49     return networkService
50       .sendRequest(target: target, parser: par
51       .eraseToAnyPublisher()
52   }
53   }{/contents}}}{{/operation}}
54   }{/operations}}
55 }
```

```
Users > azonov > Downloads > EnsService.swift > ...
15 final class EnsService: EnsServiceProtocol {
16
17   private let environment: EnvironmentConfiguratio
18   private let networkService: NetworkServiceProtoc
19   private let featureService: FeatureServiceProtoc
20
21   // MARK: Lifecycle
22
23   init(
24     environment: EnvironmentConfigurationProtoco
25     networkService: NetworkServiceProtocol,
26     featureService: FeatureServiceProtocol
27   ) {
28     self.environment = environment
29     self.networkService = networkService
30     self.featureService = featureService
31   }
32
33   // MARK: Internal
34
35   func getEnsSaldo(companyId: String) -> AnyPublis
36   let target: SMETargetType
37   let parser = CodableParser<SaldoEns>()
38   let parameters = GetEnsSaldoTarget.Parameter
39   target = GetEnsSaldoTarget(baseUrlString: en
40   return networkService
41     .sendRequest(target: target, parser: par
42     .eraseToAnyPublisher()
43   }
44
45 }
```

Строка 1, столбец 1 Пробелов: 4 UTF-8 LF {} Swift

# OpenAPI Codegen



Единый источник правды



Не тратим времени на  
бойлерплейт



Работа по-контрактам



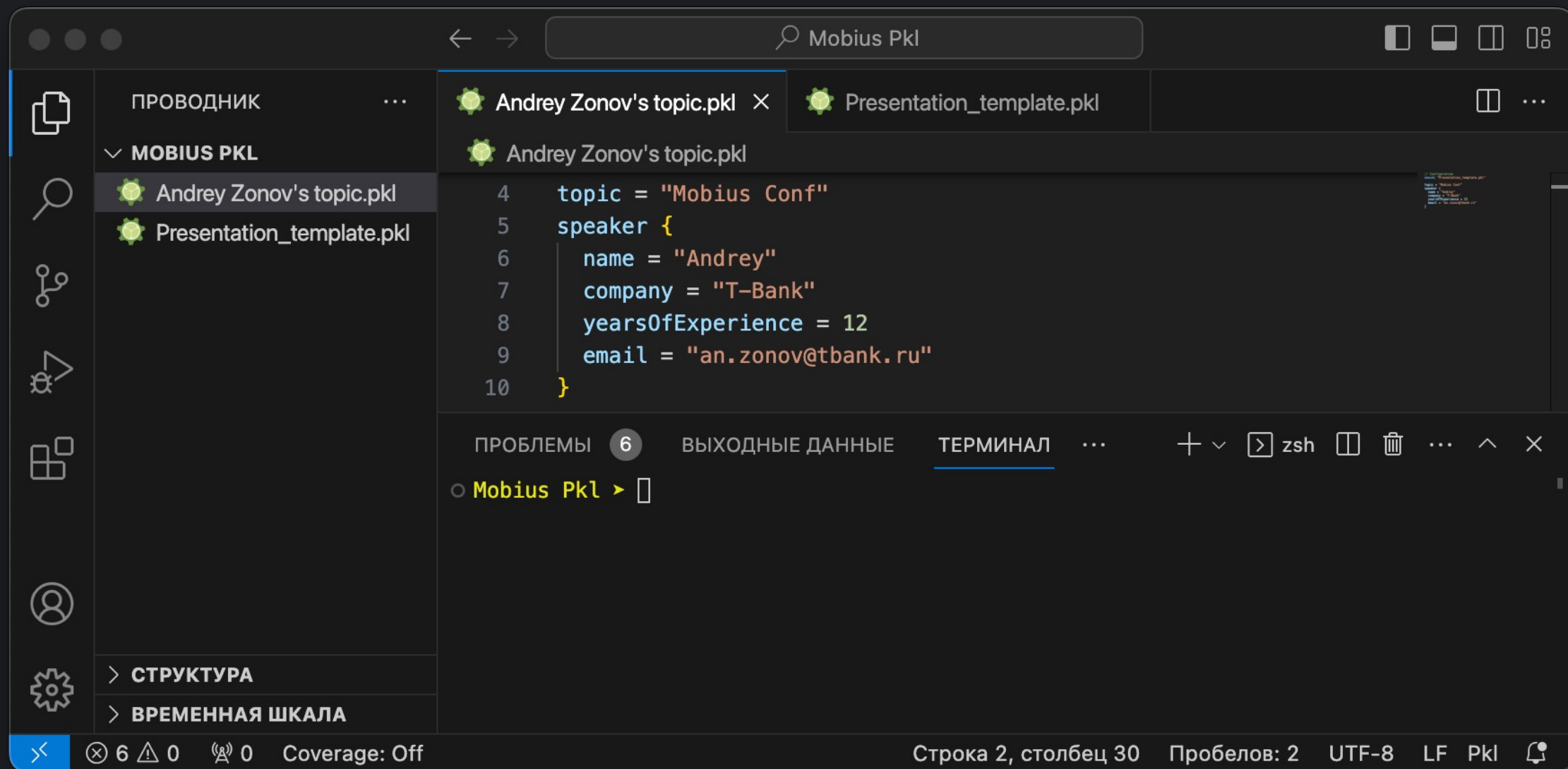
# OpenAPI Codegen

- Множество инструментов
- Не поддерживает валидацию из коробки
- Не для конфигов



**PKI**

# Генерация статической конфигурации



The screenshot shows an IDE window with the following content:

- File Explorer (left):
  - PROBODNIK
  - MOBIUS PKL
    - Andrey Zonov's topic.pkl
    - Presentation\_template.pkl
  - СТРУКТУРА
  - ВРЕМЕННАЯ ШКАЛА
- Editor (center):

```
4 topic = "Mobius Conf"
5 speaker {
6   name = "Andrey"
7   company = "T-Bank"
8   yearsOfExperience = 12
9   email = "an.zonov@tbank.ru"
10 }
```
- Terminal (bottom):
  - PROBLEMY 6
  - ВЫХОДНЫЕ ДАННЫЕ
  - ТЕРМИНАЛ
  - zsh
  - Mobius Pkl >
- Status Bar (bottom):
  - Coverage: Off
  - Строка 2, столбец 30
  - Пробелов: 2
  - UTF-8
  - LF
  - PKI

- Определяем типизированный формат конфигурации

# Генерация статической конфигурации

The screenshot shows an IDE window titled "Mobius Pkl". The left sidebar shows a file explorer with "MOBIUS PKL" containing "Andrey Zonov's topic.pkl" and "Presentation\_template.pkl". The main editor displays the PKL code for "Andrey Zonov's topic.pkl":

```
4 topic = "Mobius Conf"
5 speaker {
6   name = "Andrey"
7   company = "T-Bank"
8   yearsOfExperience = 12
9   email = "an.zonov@tbank.ru"
10 }
```

Below the editor is a terminal window with the following command and output:

```
● Mobius Pkl > pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "topic": "Mobius Conf",
  "speaker": {
    "name": "Andrey",
    "company": "T-Bank",
    "yearsOfExperience": 12,
    "email": "an.zonov@tbank.ru"
  }
}
```

The status bar at the bottom indicates "Строка 2, столбец 30 Пробелов: 2 UTF-8 LF Pkl".

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате

# Генерация статической конфигурации

The screenshot shows the Mobius Pkl IDE interface. The left sidebar contains a file explorer with a folder named 'MOBIUS PKL' containing two files: 'Andrey Zonov's topic.pkl' and 'Presentation\_template.pkl'. The main editor displays the content of 'Andrey Zonov's topic.pkl' as a PKL script:

```
4 topic = "Mobius Conf"
5 speaker {
6   name = "Andrey"
7   company = "T-Bank"
8   yearsOfExperience = 12
9   email = "an.zonov@tbank.ru"
10 }
```

Below the editor, the 'ТЕРМИНАЛ' (Terminal) tab is active, showing the command and its output:

```
Mobius Pkl > pkl eval --format yaml Andrey\ Zonov\'s\ topic.pkl
topic: Mobius Conf
speaker:
  name: Andrey
  company: T-Bank
  yearsOfExperience: 12
  email: an.zonov@tbank.ru
```

The status bar at the bottom indicates the current cursor position: 'Строка 2, столбец 30' (Line 2, Column 30), along with other settings like 'Пробелов: 2', 'UTF-8', 'LF', and 'Pkl'.

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате

# Генерация статической конфигурации

The screenshot shows an IDE window titled "Mobius Pkl". The editor displays a PKL file named "Andrey Zonov's topic.pkl" with the following content:

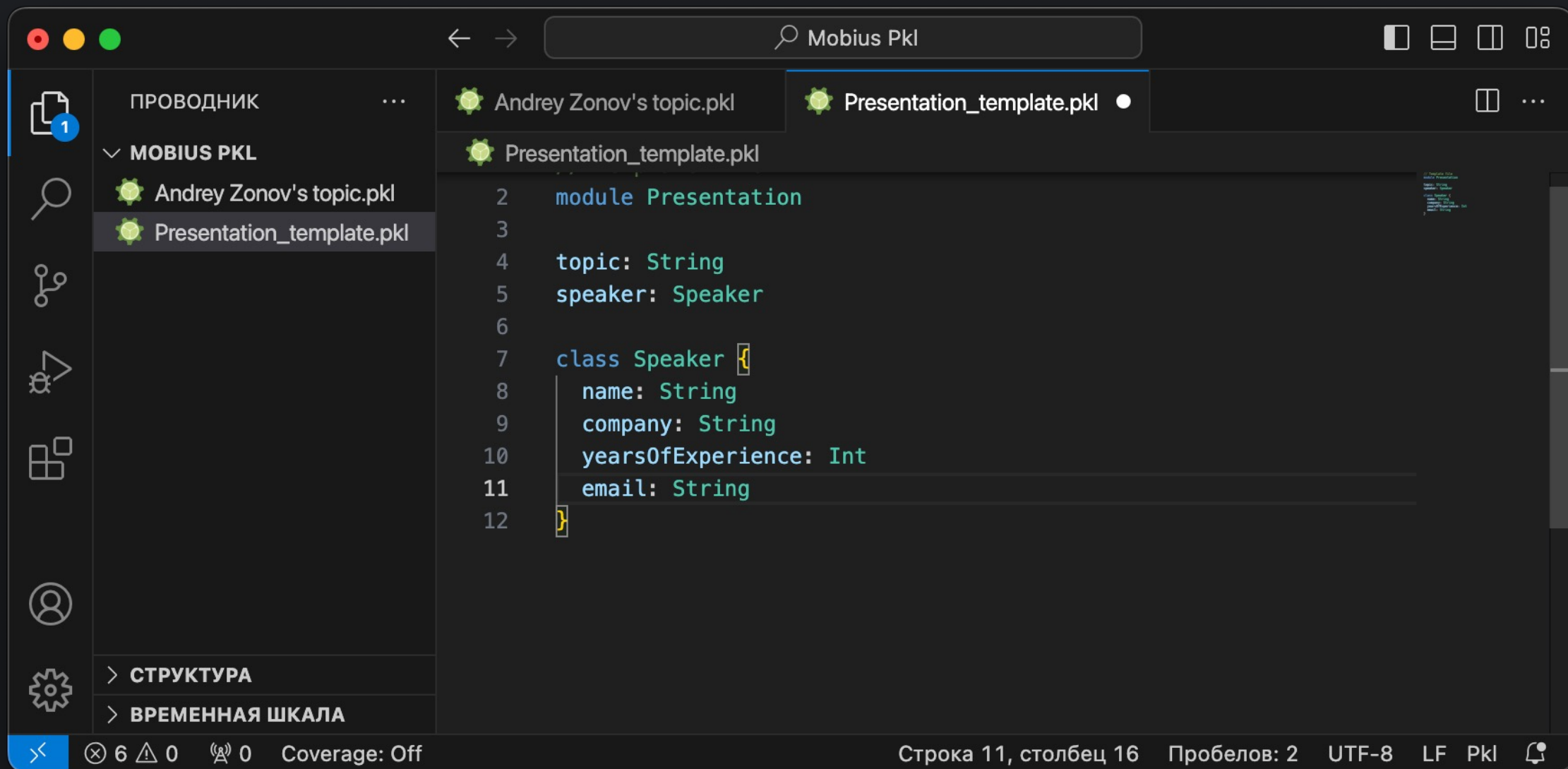
```
4 topic = "Mobius Conf"
5 speaker {
6   name = "Andrey"
7   company = "T-Bank"
8   yearsOfExperience = 12
9   email = "an.zonov@tbank.ru"
10 }
```

Below the editor, the terminal shows the command and its output:

```
Mobius Pkl > pkl eval --format xml Andrey\ Zonov\'s\ topic.pkl
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <topic>Mobius Conf</topic>
  <speaker>
    <name>Andrey</name>
    <company>T-Bank</company>
    <yearsOfExperience>12</yearsOfExperience>
    <email>an.zonov@tbank.ru</email>
  </speaker>
```

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате

# Генерация статической конфигурации

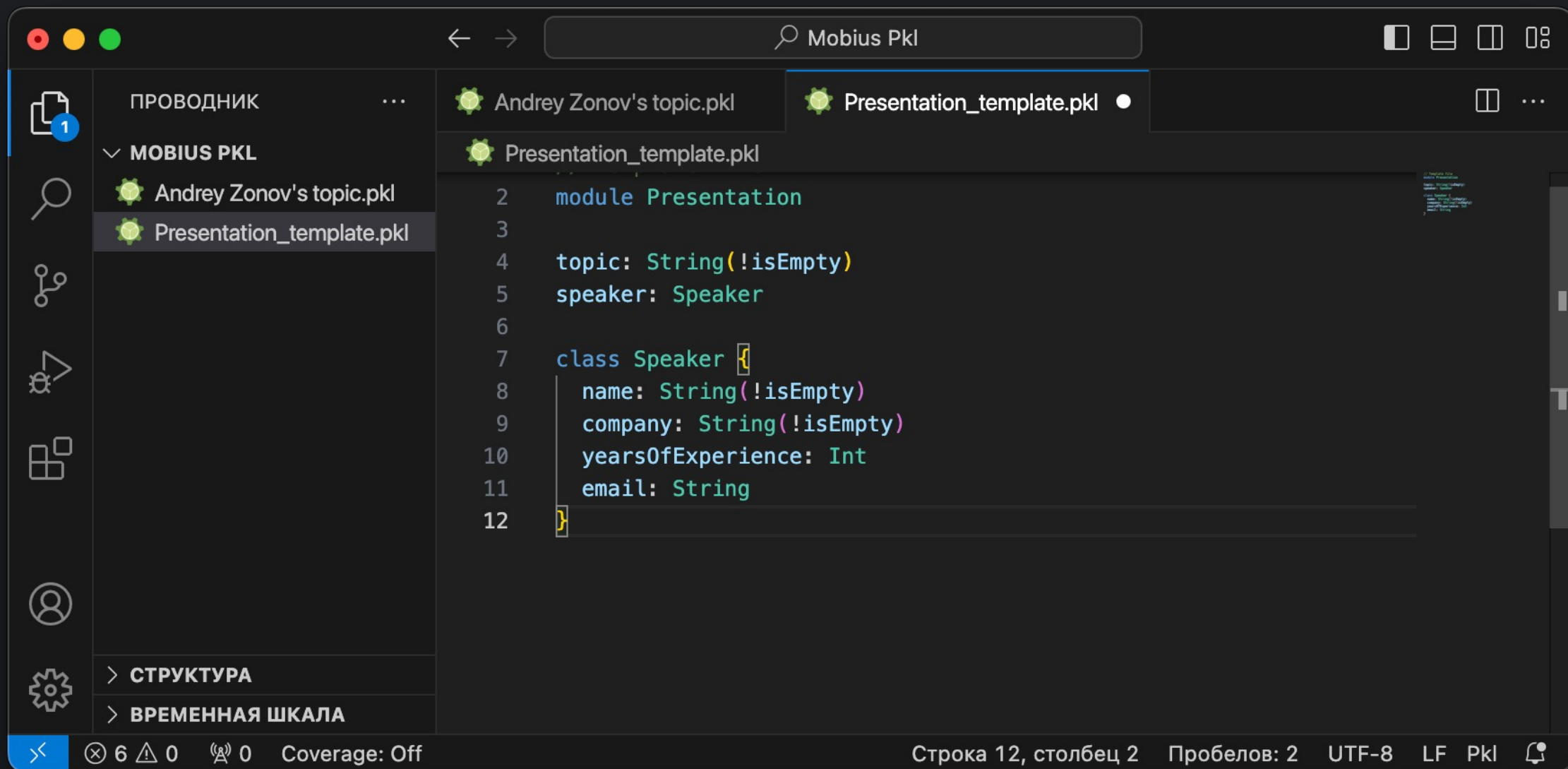


```
2  module Presentation
3
4  topic: String
5  speaker: Speaker
6
7  class Speaker {
8    name: String
9    company: String
10   yearsOfExperience: Int
11   email: String
12 }
```

The screenshot shows an IDE window titled 'Mobius Pkl'. The left sidebar shows a file explorer with a folder 'MOBIUS PKL' containing two files: 'Andrey Zonov's topic.pkl' and 'Presentation\_template.pkl'. The main editor displays the code for 'Presentation\_template.pkl'. The code defines a module 'Presentation' with two fields: 'topic' of type 'String' and 'speaker' of type 'Speaker'. The 'Speaker' class has four fields: 'name' (String), 'company' (String), 'yearsOfExperience' (Int), and 'email' (String). The status bar at the bottom indicates 'Строка 11, столбец 16 Пробелов: 2 UTF-8 LF Pkl'.

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате
- Определяем темплейт для данных

# Генерация статической конфигурации



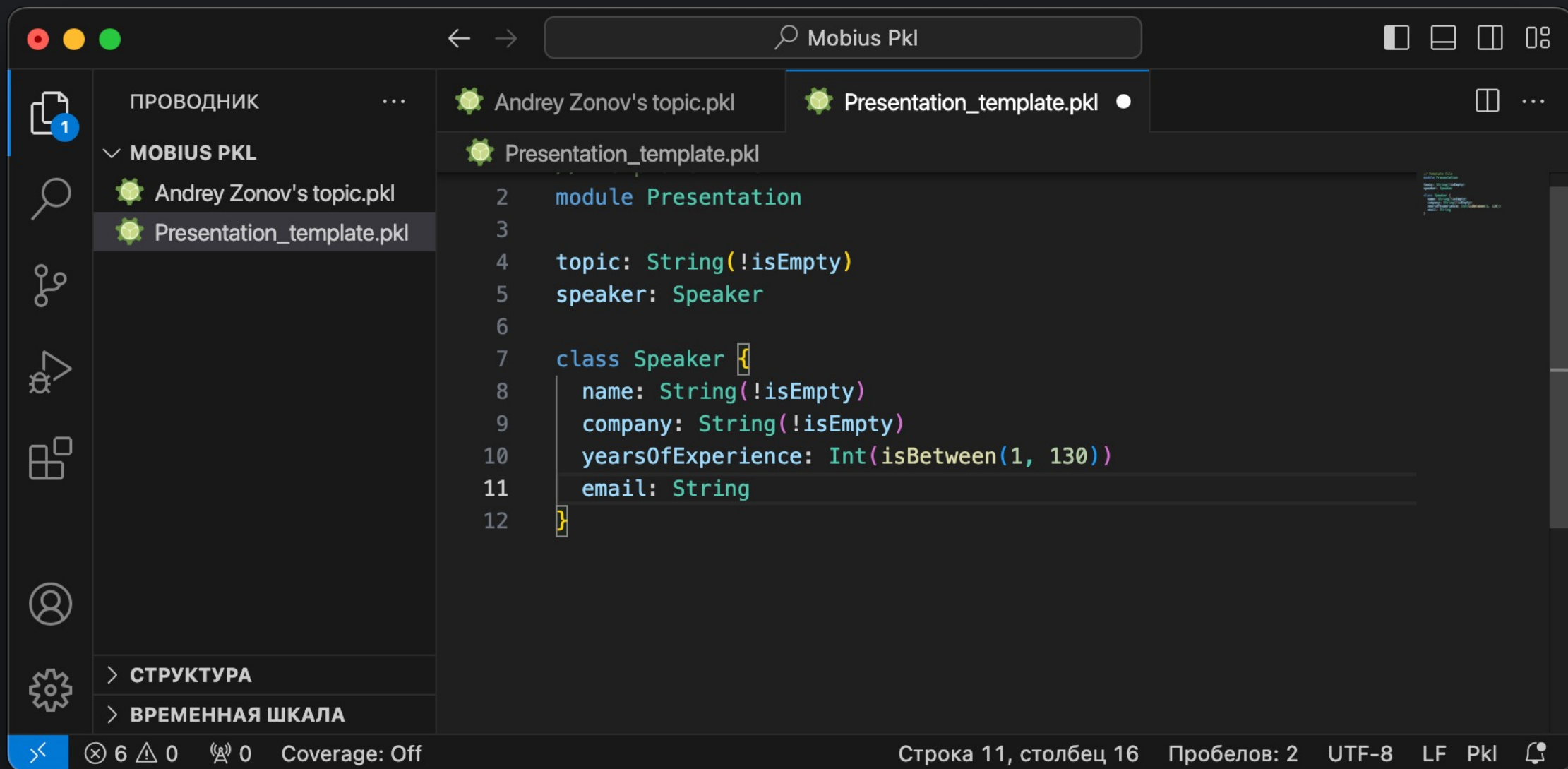
```
2 module Presentation
3
4 topic: String(!isEmpty)
5 speaker: Speaker
6
7 class Speaker {
8   name: String(!isEmpty)
9   company: String(!isEmpty)
10  yearsOfExperience: Int
11  email: String
12 }
```

Строка 12, столбец 2 Пробелов: 2 UTF-8 LF Pkl

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате
- Определяем темплейт для данных
- Задаем ограничения



# Генерация статической конфигурации

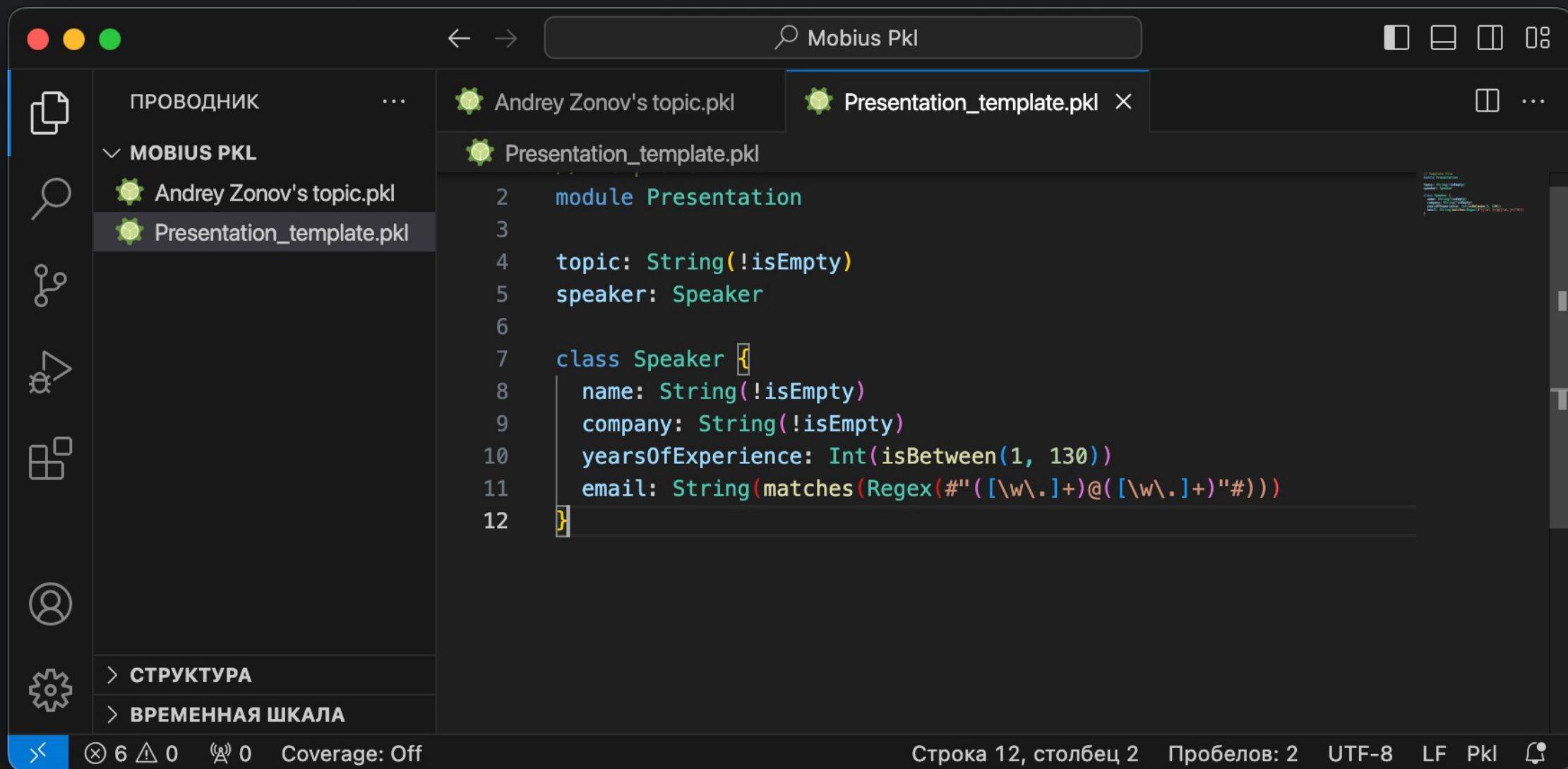


```
2 module Presentation
3
4 topic: String(!isEmpty)
5 speaker: Speaker
6
7 class Speaker {
8   name: String(!isEmpty)
9   company: String(!isEmpty)
10  yearsOfExperience: Int(isBetween(1, 130))
11  email: String
12 }
```

The screenshot shows an IDE window titled 'Mobius Pkl'. The left sidebar shows a file explorer with 'MOBIUS PKL' containing 'Andrey Zonov's topic.pkl' and 'Presentation\_template.pkl'. The main editor displays the PKL code for 'Presentation\_template.pkl'. The code defines a 'Presentation' module with a 'topic' field of type 'String(!isEmpty)' and a 'speaker' field of type 'Speaker'. The 'Speaker' class has fields: 'name' (String(!isEmpty)), 'company' (String(!isEmpty)), 'yearsOfExperience' (Int(isBetween(1, 130))), and 'email' (String). The status bar at the bottom indicates 'Строка 11, столбец 16 Пробелов: 2 UTF-8 LF Pkl'.

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате
- Определяем темплейт для данных
- Задаем ограничения

# Генерация статической конфигурации



```
2 module Presentation
3
4 topic: String(!isEmpty)
5 speaker: Speaker
6
7 class Speaker {
8   name: String(!isEmpty)
9   company: String(!isEmpty)
10  yearsOfExperience: Int(isBetween(1, 130))
11  email: String(matches(Regex("#"([\w\.]+"#"))))
12 }
```

Строка 12, столбец 2 Пробелов: 2 UTF-8 LF Pkl

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате
- Определяем темплейт для данных
- Задаем ограничения

# Генерация статической конфигурации

The screenshot shows an IDE window titled "Mobius Pk1". The left sidebar shows a file explorer with "MOBIUS PKL" containing "Andrey Zonov's topic.pk1" and "Presentation\_template.pk1". The main editor displays the content of "Andrey Zonov's topic.pk1":

```
4 topic = "Mobius Conf"
5 speaker {
6   name = "Andrey"
7   company = "T-Bank"
8   yearsOfExperience = 0
9   email = "an.zonov@tbank.ru"
10 }
```

The terminal window shows the command `pk1 eval Andrey\ Zonov\'s\ topic.pk1` and the resulting error:

```
⊗ Mobius Pk1 > pk1 eval Andrey\ Zonov\'s\ topic.pk1
— Pk1 Error —
Type constraint `isBetween(1, 130)` violated.
Value: 0

10 | yearsOfExperience: Int(isBetween(1, 130))
    | ~~~~~
at Presentation#Speaker.yearsOfExperience (file:///Users/azonov/Developer/Mobius%20Pk1/Presentation_template.pk1, line 10)
```

The status bar at the bottom indicates "Строка 8, столбец 24 Пробелов: 2 UTF-8 LF Pk1".

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате
- Определяем темплейт для данных
- Задаем ограничения
- Валидируем

# Генерация статической конфигурации

The screenshot shows the Mobius Pk1 IDE interface. The left sidebar contains a file explorer with a folder named 'MOBIUS PKL' containing two files: 'Andrey Zonov's topic.pkl' and 'Presentation\_template.pkl'. The main editor displays the content of 'Andrey Zonov's topic.pkl' with the following code:

```
2 amends "Presentation_template.pkl"
3
4 topic = "Mobius Conf"
5 speaker {
6   name = "Andrey"
7   company = "T-Bank"
8   yearsOfExperience = 12
}
```

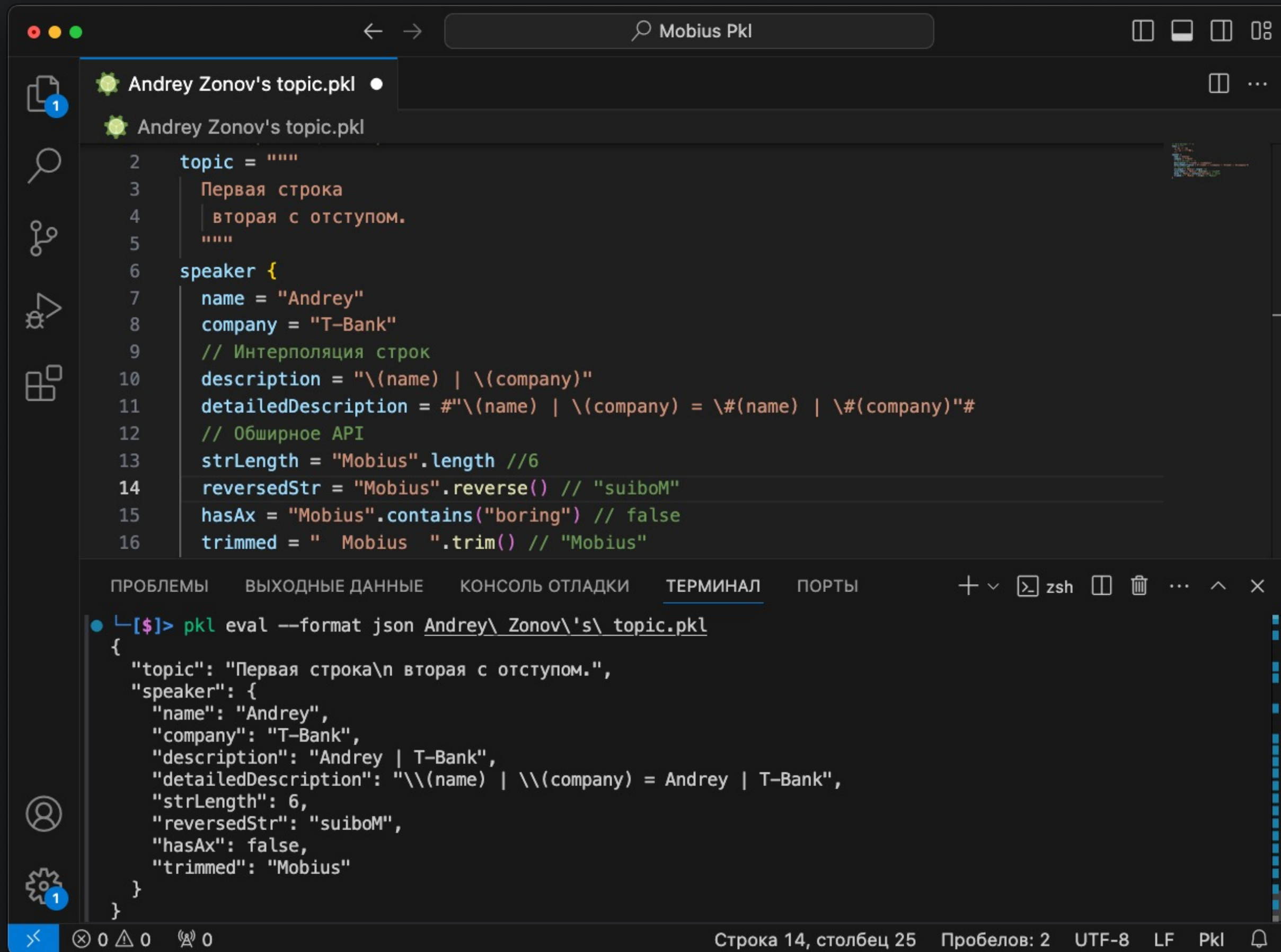
Below the editor is a terminal window with the following output:

```
● Mobius Pk1 > pkl eval Andrey\ Zonov\'s\ topic.pkl
topic = "Mobius Conf"
speaker {
  name = "Andrey"
  company = "T-Bank"
  yearsOfExperience = 12
  email = "an.zonov@tbank.ru"
}
○ Mobius Pk1 > []
```

The status bar at the bottom indicates 'Строка 6, столбец 18 Пробелов: 2 UTF-8 LF PK1'.

- Определяем типизированный формат конфигурации
- Генерируем данные для транспортного уровня в любом удобном формате
- Определяем темплейт для данных
- Задаем ограничения
- Валидируем

# ВОЗМОЖНОСТИ ЯЗЫКА



The screenshot shows the Pkl IDE interface. The top part displays the source file 'Andrey Zonov's topic.pkl' with the following code:

```
2 topic = ""
3   Первая строка
4   | вторая с отступом.
5   ""
6 speaker {
7   name = "Andrey"
8   company = "T-Bank"
9   // Интерполяция строк
10  description = "\(name) | \(company)"
11  detailedDescription = "#\(name) | \(company) = \(name) | \(company)"#
12  // Обширное API
13  strLength = "Mobius".length //6
14  reversedStr = "Mobius".reverse() // "suiboM"
15  hasAx = "Mobius".contains("boring") // false
16  trimmed = " Mobius ".trim() // "Mobius"
```

The bottom part shows a terminal window with the command and its output:

```
└─[$]> pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "topic": "Первая строка\n вторая с отступом.",
  "speaker": {
    "name": "Andrey",
    "company": "T-Bank",
    "description": "Andrey | T-Bank",
    "detailedDescription": "\\(name) | \\(company) = Andrey | T-Bank",
    "strLength": 6,
    "reversedStr": "suiboM",
    "hasAx": false,
    "trimmed": "Mobius"
  }
}
```

The status bar at the bottom indicates: Строка 14, столбец 25 Пробелов: 2 UTF-8 LF Pkl

- Удобная работа со строками

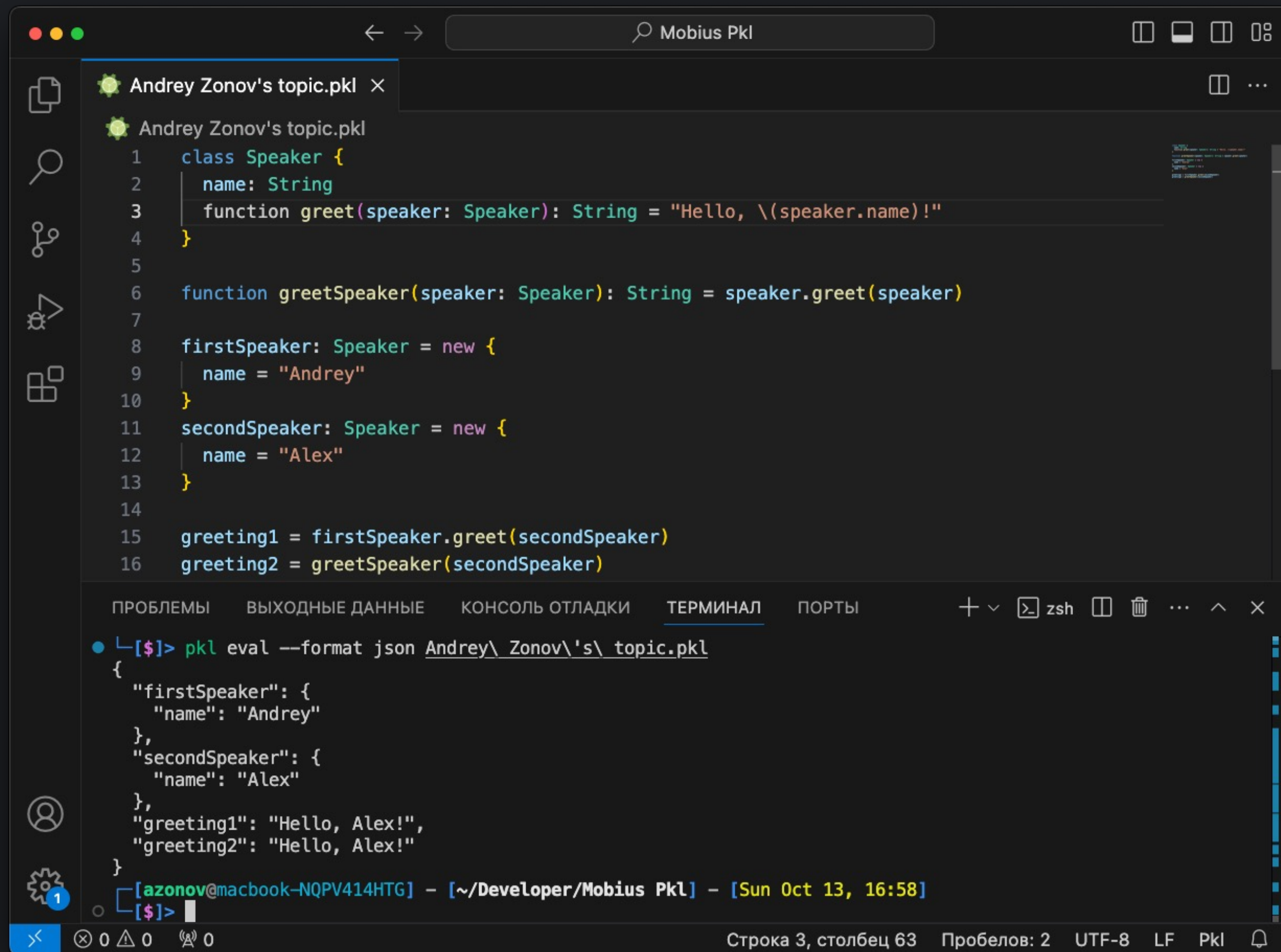
# ВОЗМОЖНОСТИ ЯЗЫКА

```
Andrey Zonov's topic.pki
Users > azonov > Developer > Mobius Pki > Andrey Zonov's topic.pki
1  name: String(!isBlank)
2
3  email: String(contains("@"))
4
5  swiftFile: String(endsWith(".swift"))
6
7  contentType: "text/plain"|"text/html"|"application/json"|String
8
```

Строка 8, столбец 1 Пробелов: 2 UTF-8 LF PKI

- Удобная работа со строками

# ВОЗМОЖНОСТИ ЯЗЫКА



The screenshot shows a code editor window with a file named "Andrey Zonov's topic.pkl". The code defines a class `Speaker` with a `greet` function, a `greetSpeaker` function, and two instance variables `firstSpeaker` and `secondSpeaker`. Below the code, a terminal window shows the command `pkl eval --format json Andrey\ Zonov\'s\ topic.pkl` and its output, which is a JSON object containing the speaker names and their respective greetings.

```
1 class Speaker {
2   name: String
3   function greet(speaker: Speaker): String = "Hello, \(speaker.name)!"
4 }
5
6 function greetSpeaker(speaker: Speaker): String = speaker.greet(speaker)
7
8 firstSpeaker: Speaker = new {
9   name = "Andrey"
10 }
11 secondSpeaker: Speaker = new {
12   name = "Alex"
13 }
14
15 greeting1 = firstSpeaker.greet(secondSpeaker)
16 greeting2 = greetSpeaker(secondSpeaker)
```

```
PROБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ  ПОРТЫ
└─[$]> pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "firstSpeaker": {
    "name": "Andrey"
  },
  "secondSpeaker": {
    "name": "Alex"
  },
  "greeting1": "Hello, Alex!",
  "greeting2": "Hello, Alex!"
}
[azonov@macbook-NQPV414HTG] - [~/Developer/Mobius Pkl] - [Sun Oct 13, 16:58]
└─[$]>
```

Строка 3, столбец 63 Пробелов: 2 UTF-8 LF Pkl

- Удобная работа со строками
- ООП

# ВОЗМОЖНОСТИ ЯЗЫКА

The screenshot shows an IDE window titled "Mobius Pk1". The main editor displays the following Kotlin code:

```
1 port = read?("env:PORT")?.toInt() ?? 1234
2 name = "Andrey"
3 nameLength = name?.length
4 nameUpper = name?.toUpperCase()
5
6 name2 = null
7 name2Length = name2?.length
8 name2Upper = name2?.toUpperCase()
```

Below the code is a terminal window with the following command and output:

```
└─[$]> pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "port": 1234,
  "name": "Pigeon",
  "nameLength": 6,
  "nameUpper": "PIGEON"
}
```

The status bar at the bottom indicates: "Строка 5, столбец 1 Пробелов: 2 UTF-8 LF Pk1".

- Удобная работа со строками
- ООП
- Nullable



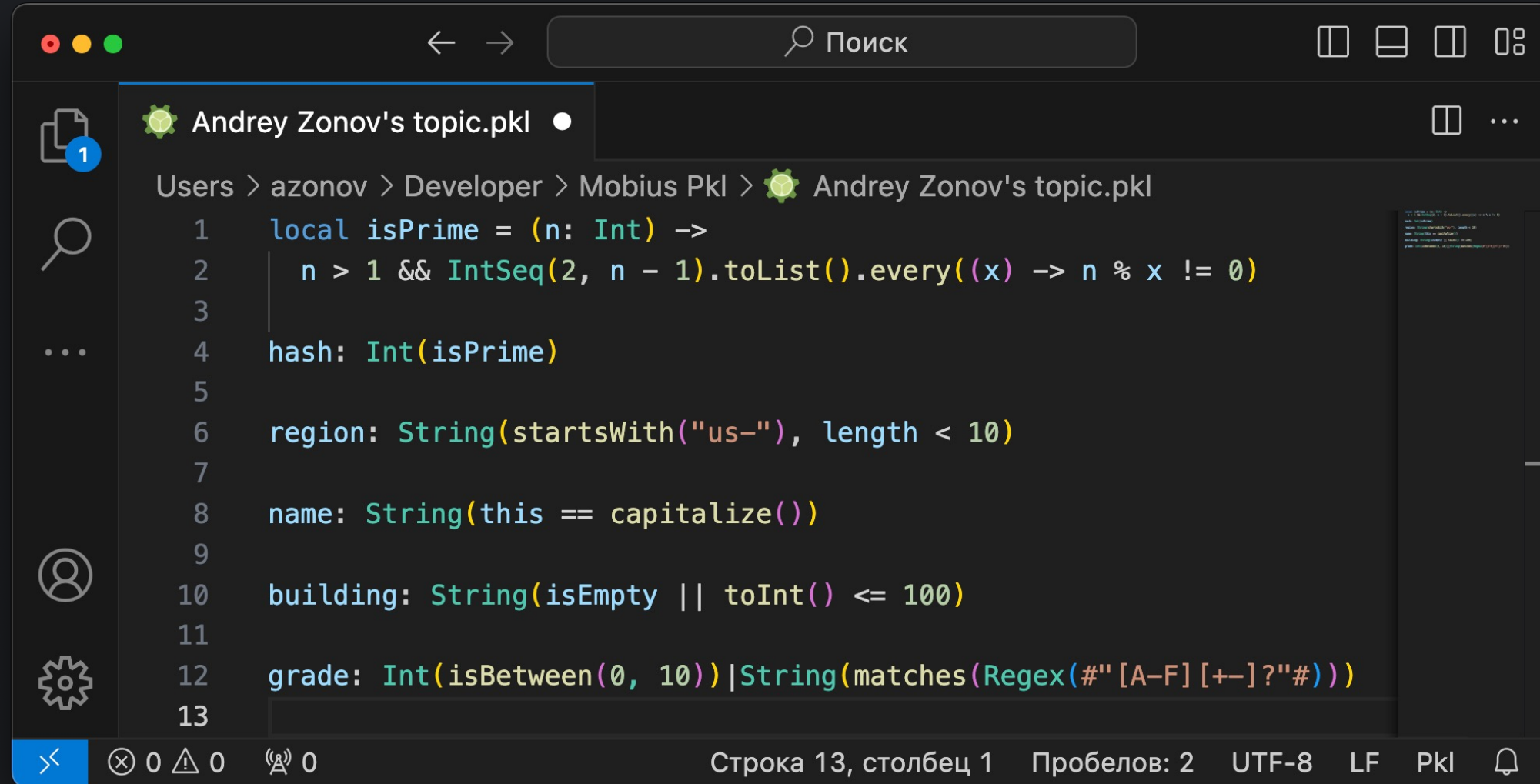
# ВОЗМОЖНОСТИ ЯЗЫКА

```
Andrey Zonov's topic.pkl x
Andrey Zonov's topic.pkl
1 list = List(1, 2, 3, 4)
2 res1l = list.contains(3)
3 res2l = list.drop(1).take(2)
4 res3l = list.map((n) -> n * 3)
5
6 set = Set(1, 2, 3, 4)
7 res1s = set.contains(3)
8 res2s = set.drop(1).take(2)
9 res3s = set.map((n) -> n * 3)
10 res4s = set.intersect(Set(3, 9, 2))
11
12 map = Map("Pigeon", 5.gb, "Parrot", 10.gb)
13 res1m = map.containsKey("Parrot")
14 res2m = map.containsValue(8.gb)
15 res3m = map.getOrElse("Falcon", null)

ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  ТЕРМИНАЛ  ...  + v zsh
└─┬[$]─> pkl eval Andrey\ Zonov\'s\ topic.pkl
list = List(1, 2, 3, 4)
res1l = true
res2l = List(2, 3)
res3l = List(3, 6, 9, 12)
set = Set(1, 2, 3, 4)
res1s = true
res2s = Set(2, 3)
res3s = Set(3, 6, 9, 12)
res4s = Set(3, 2)
map = Map("Pigeon", 5.gb, "Parrot", 10.gb)
res1m = true
res2m = false
res3m = null
[azonov@macbook-NQPV414HTG] - [~/Developer/Mobius Pkl] - [Sun Oct 13, 17:07]
└─┬[$]─>
```

- Удобная работа со строками
- ООП
- Nullable
- Коллекции

# ВОЗМОЖНОСТИ ЯЗЫКА



```
1 local isPrime = (n: Int) ->
2   n > 1 && IntSeq(2, n - 1).toList().every((x) -> n % x != 0)
3
4 hash: Int(isPrime)
5
6 region: String(startsWith("us-"), length < 10)
7
8 name: String(this == capitalize())
9
10 building: String(isEmpty || toInt() <= 100)
11
12 grade: Int(isBetween(0, 10)) | String(matches(Regex("#[A-F][+-]?"#)))
13
```

- Удобная работа со строками
- ООП
- Nullable
- Коллекции
- Ограничения

# Кодогенерация для популярных ЯП

```
PKI      Java      Kotlin      Swift      Go
-----
module example.MyAppConfig

/// The hostname for the application
host: String

/// The port to listen on
port: UInt16(this > 1000)
```

- Java
- Kotlin
- Swift
- Go

```
PKI      Java      Kotlin
-----
package example;

public final class MyAppConfig {
    /**
     * The hostname for the application
     */
    public final @NonNull String host;

    /**
     * The port to listen on
     */
}
```

```
PKI      Java      Kotlin
-----
package example

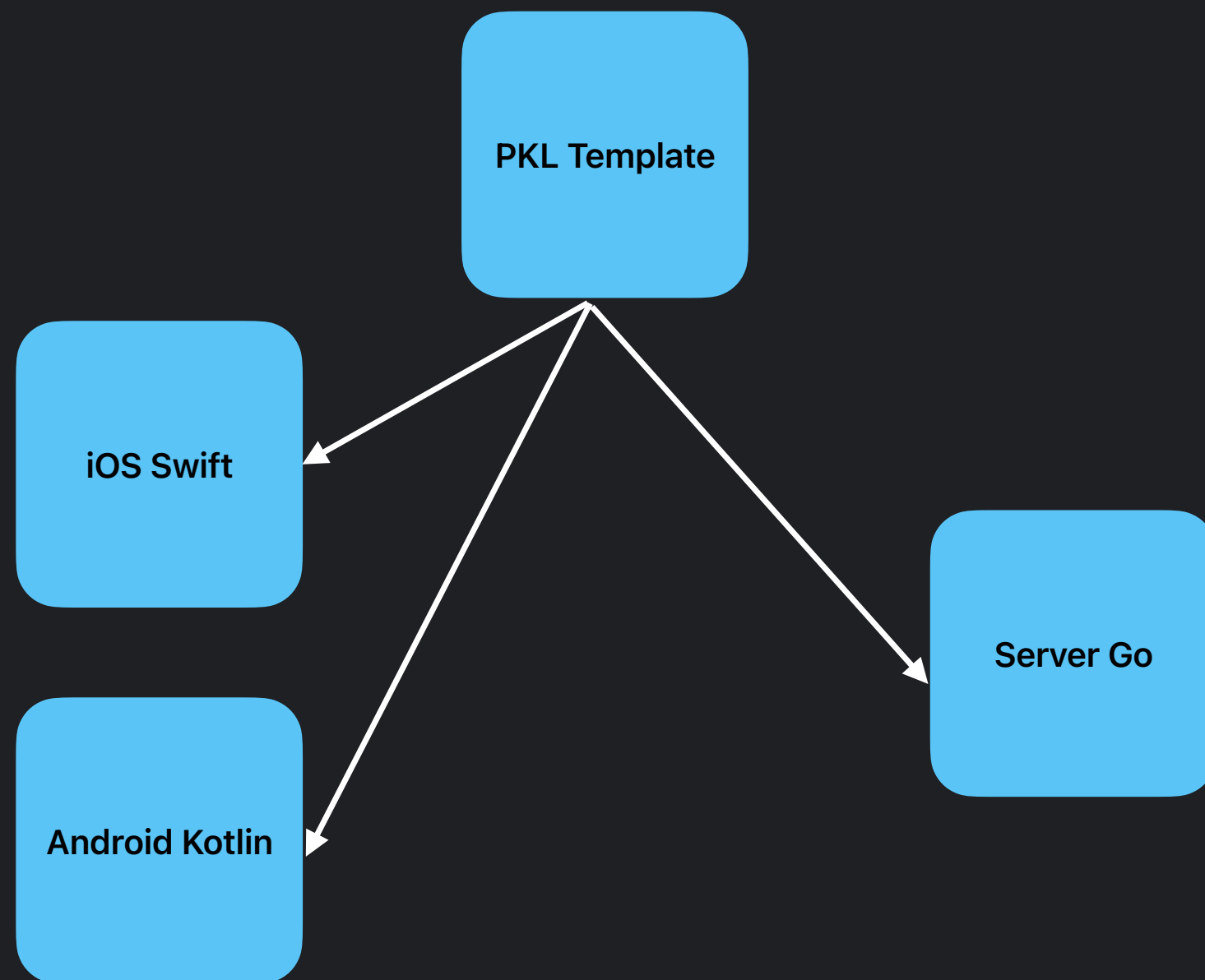
data class MyAppConfig(
    /**
     * The hostname for the applica
     */
    val host: String,
    /**
     * The port to listen on
     */
    val port: Int
)
```

```
PKI      Java      Kotlin      Swift      Go
-----
enum MyAppConfig {}

extension MyAppConfig {
    struct Module {
        /// The hostname for the application
        let host: String

        /// The port to listen on
        let port: UInt16
    }
}
```

# Кодогенерация для популярных ЯП



- Java
- Kotlin
- Swift
- Go

# Кодогенерация

Generated > Presentation.pkl.swift > ...

```
1 // Code generated from Pkl module `Presentation`. DO NOT EDIT.
2 import PklSwift
3
4 public enum Presentation {}
5
6 extension Presentation {
7     public struct Module: PklRegisteredType, Decodable, Hashable {
8         public static var registeredIdentifier: String = "Presentation"
9
10        public var topic: String
11
12        public var speaker: Speaker
13
14        public init(topic: String, speaker: Speaker) {
15            self.topic = topic
16            self.speaker = speaker
17        }
18    }
19
20    public struct Speaker: PklRegisteredType, Decodable, Hashable {
21        public static var registeredIdentifier: String = "Presentation#"
22
23        public var name: String
24
25        public var company: String
26
27        public var yearsOfExperience: Int
28
29        public var email: String
```

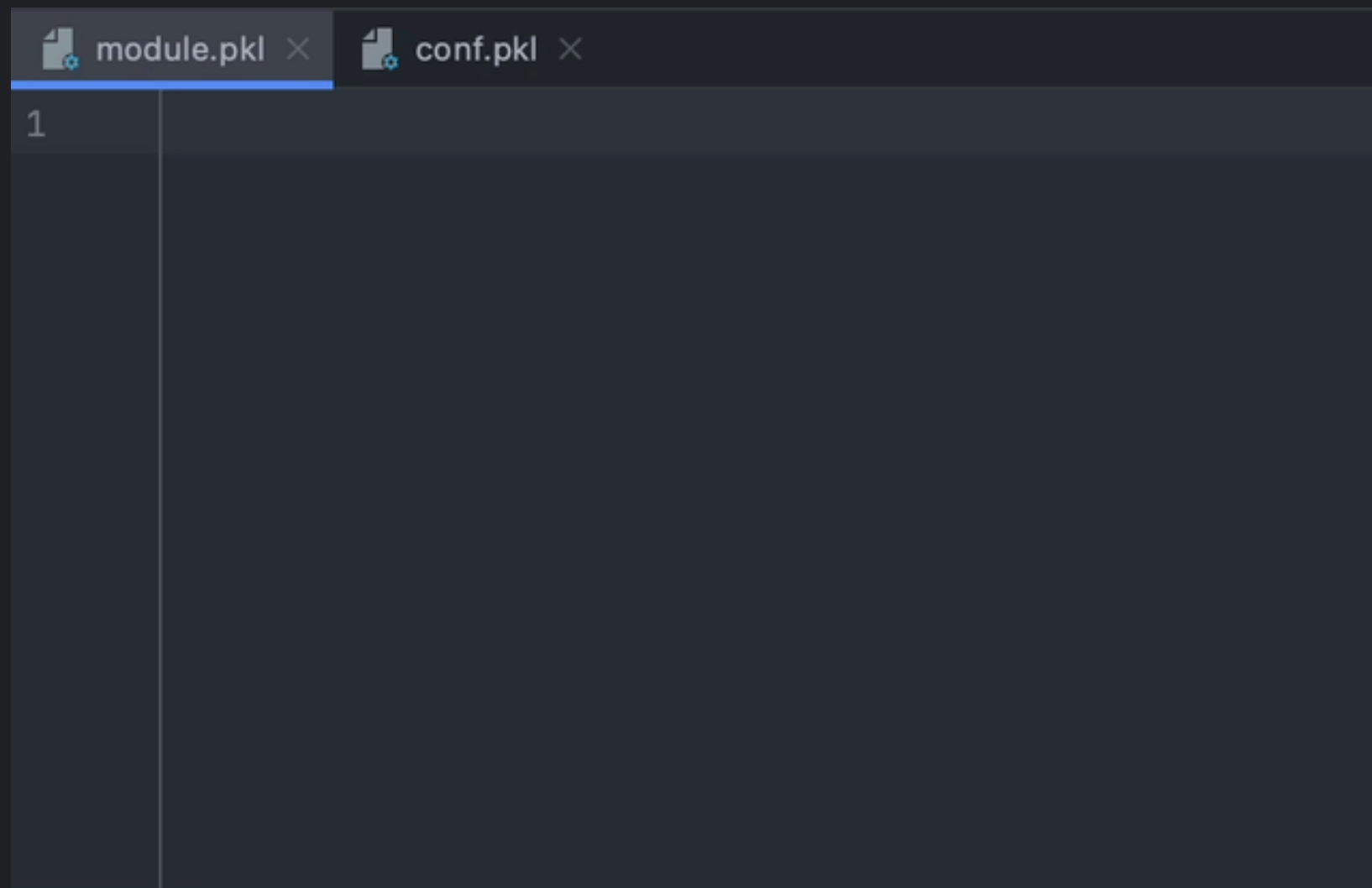
ПРОБЛЕМЫ 6 Выходные данные ТЕРМИНАЛ ... + v zsh ... ^ x

● Mobius Pkl > ~/.pkl-gen-swift Andrej\ Zonov\'s\ topic.pkl -o Generated/  
/Users/azonov/Developer/Mobius Pkl/Generated/Presentation.pkl.swift  
○ Mobius Pkl > |

Строка 1, столбец 1 Пробелов: 4 UTF-8 LF {} Swift

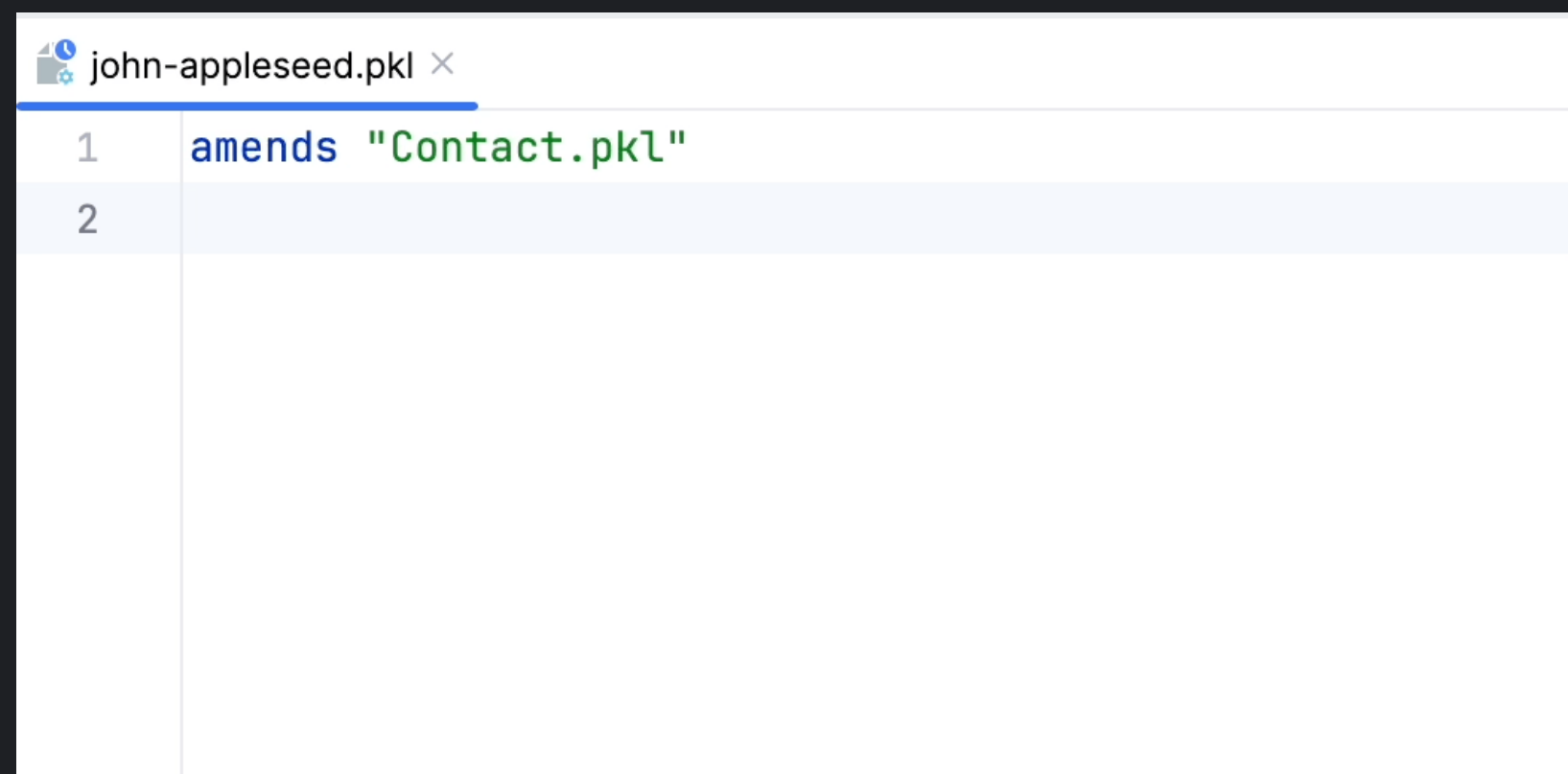
# Интеграция с IDE через LSP

- IntelliJ
- VS Code
- Neovim
- Xcode



# Валидация

- IntelliJ
- VS Code
- Neovim
- Xcode



The screenshot shows a code editor window with a single tab titled "john-appleseed.pkl". The editor contains two lines of code. The first line is "amends 'Contact.pkl'", where "amends" is in blue and the string is in green. The second line is empty. The line numbers 1 and 2 are visible in the left margin.

```
1 amends "Contact.pkl"  
2
```

# Навигация

- IntelliJ
- VS Code
- Neovim
- Xcode

```
import "package://pkg.pkl-lang.org/pantry/temp.io.k8s@1.0.0#/K8sObject.pkl"
```

```
quant: K8sObject.Quantity
```

```
|
```



# АВТОКОМПЛИТ

- IntelliJ
- VS Code
- Neovim
- Xcode

```
7 ["home"] = "+1 (555) 555-5555"  
8 }  
9 |
```

# Тестирование

The screenshot shows an IDE window titled "Mobius Pkl". The editor displays a PKL test file named "presentation\_test.pkl" with the following code:

```
1 amends "pkl:test"
2 import "package://pkg.pkl-lang.org/pkl-pantry/pkl.ex
3
4 facts {
5   ["encode"] {
6     URI.encode("https://example.com/some path") ==
7       "https://example.com/some%20path"
8     local safeChars = "!#$%&'()*+,-./:;=?@~"
9     URI.encode(safeChars) == safeChars
10    URI.encode("\u{ffff}") == "%EF%BF%BF"
11    URI.encode("🍌") == "%F0%9F%8F%80"
12  }
13  ["percentDecode"] {
14    URI.percentDecode("foo%20bar") == "foo bar"
15    URI.percentDecode("%EF%BF%BF") == "\u{ffff}"
16    URI.percentDecode("%F0%9F%8F%80") == "🍌"
17  }
18 }
```

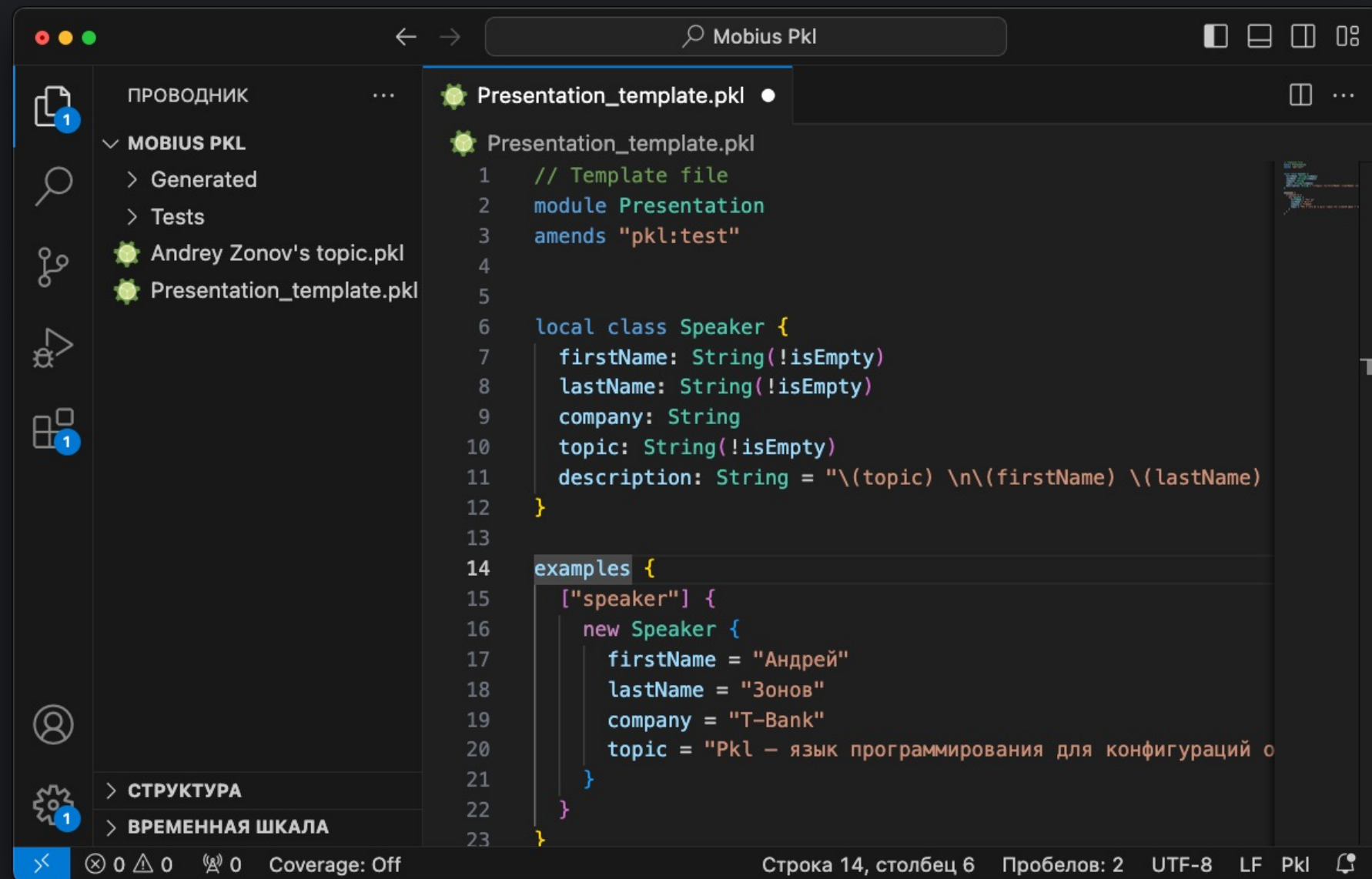
Below the editor is a terminal window showing the execution of the test:

```
PROБЛЕМЫ  ТЕРМИНАЛ  ...  + v  zsh  [ ]  [ ]  ...  ^  x
• [Mobius Pkl] pkl test presentation_test.pkl
  module presentation_test (file:///Users/azonov/Developer/Mobius%20Pkl/presentation_test.pkl, line 1)
    encode ✓
    percentDecode ✓
  ○ [Mobius Pkl] |
```

The status bar at the bottom indicates "Строка 11, столбец 38 Пробелов: 4 UTF-8 LF Pkl".

- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов

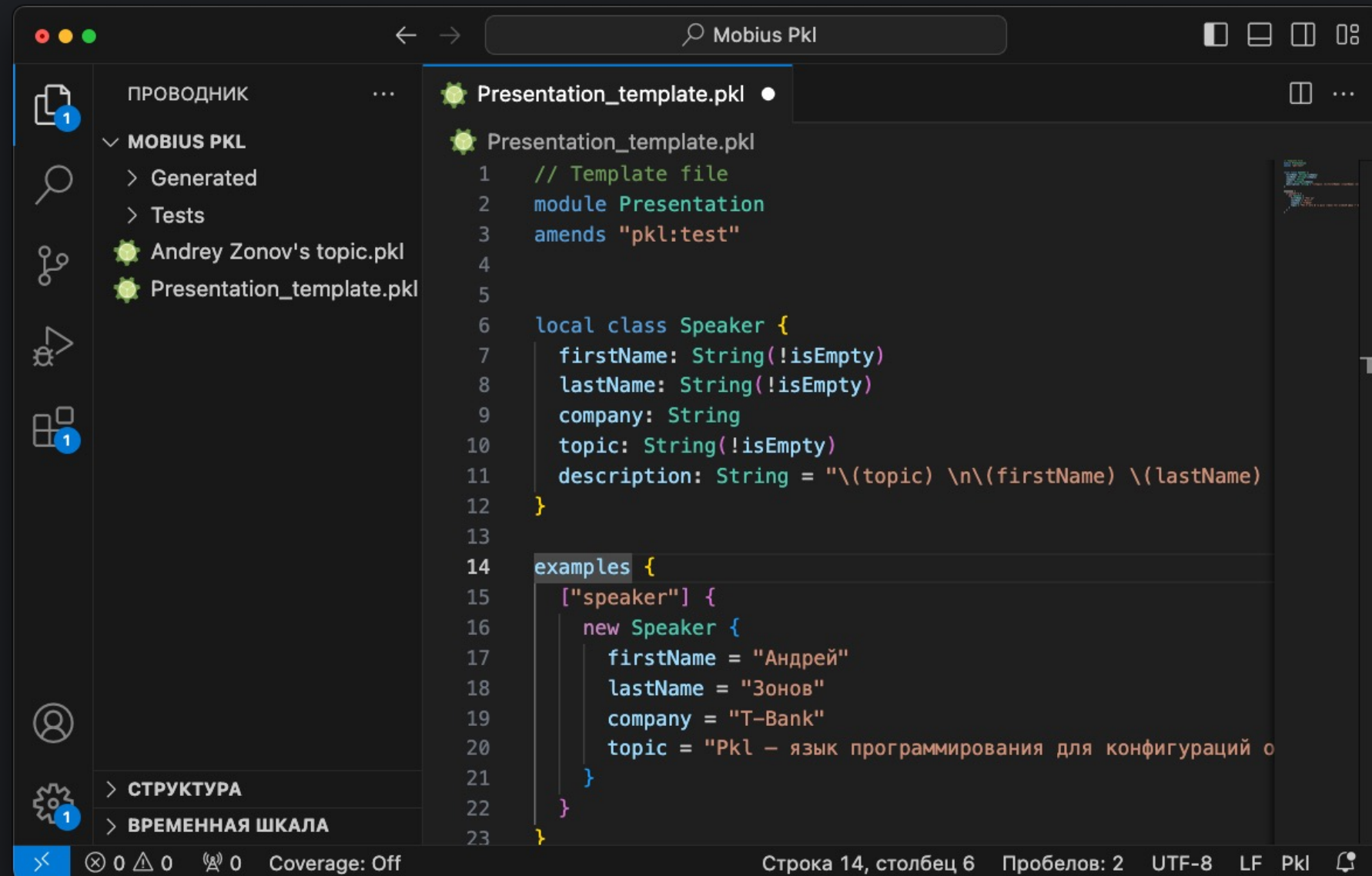
# Тестирование



```
1 // Template file
2 module Presentation
3   amends "pkl:test"
4
5
6   local class Speaker {
7     firstName: String(!isEmpty)
8     lastName: String(!isEmpty)
9     company: String
10    topic: String(!isEmpty)
11    description: String = "\(\topic) \n\(\firstName) \(\lastName)
12  }
13
14  examples {
15    ["speaker"] {
16      new Speaker {
17        firstName = "Андрей"
18        lastName = "Зонов"
19        company = "Т-Банк"
20        topic = "Pkl – язык программирования для конфигураций о
21      }
22    }
23  }
```

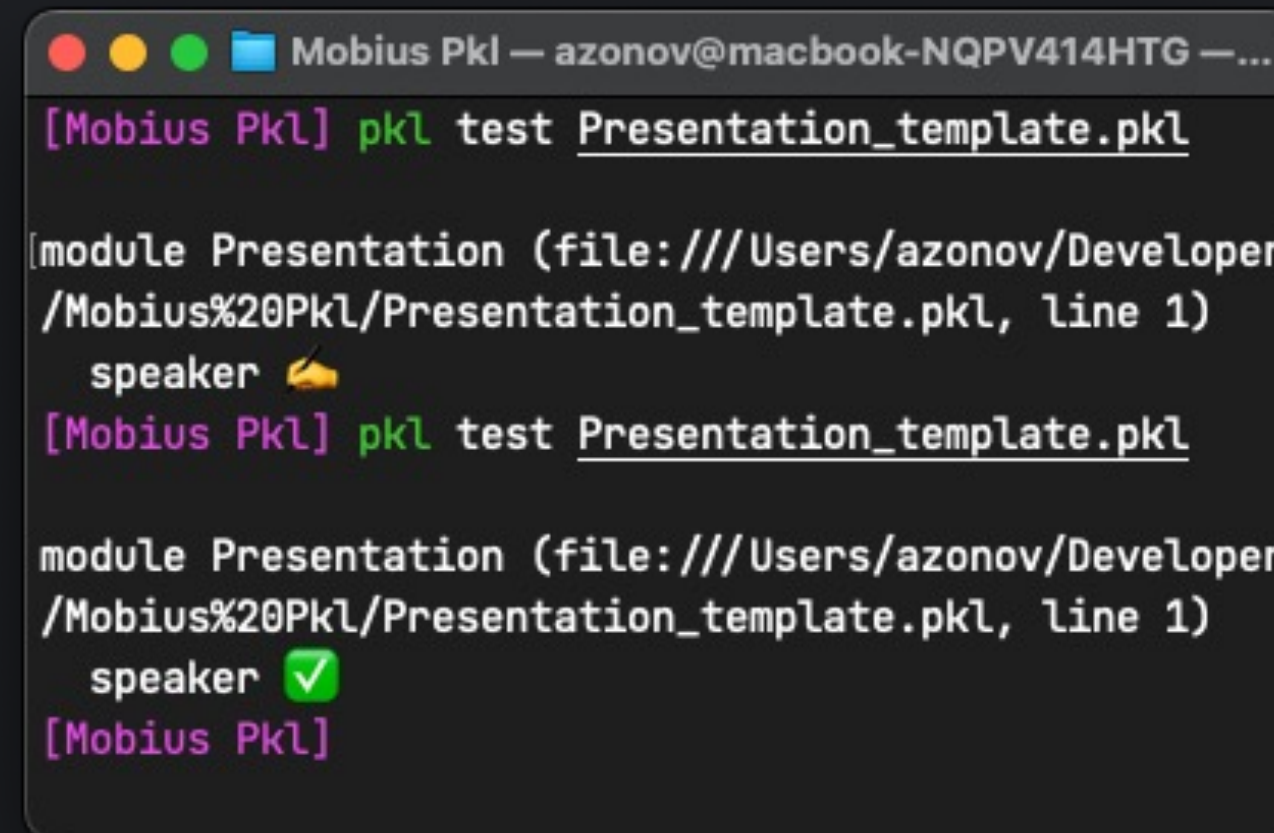
- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов
- Example – аналог снимок тестирования для конфига

# Тестирование



```
1 // Template file
2 module Presentation
3   amends "pkl:test"
4
5
6 local class Speaker {
7   firstName: String(!isEmpty)
8   lastName: String(!isEmpty)
9   company: String
10  topic: String(!isEmpty)
11  description: String = "\\(topic) \\n\\(firstName) \\(lastName)
12 }
13
14 examples {
15   ["speaker"] {
16     new Speaker {
17       firstName = "Андрей"
18       lastName = "Зонов"
19       company = "Т-Банк"
20       topic = "Pkl - язык программирования для конфигураций о
21     }
22   }
23 }
```

- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов
- Example – аналог снимот тестирования для конфига

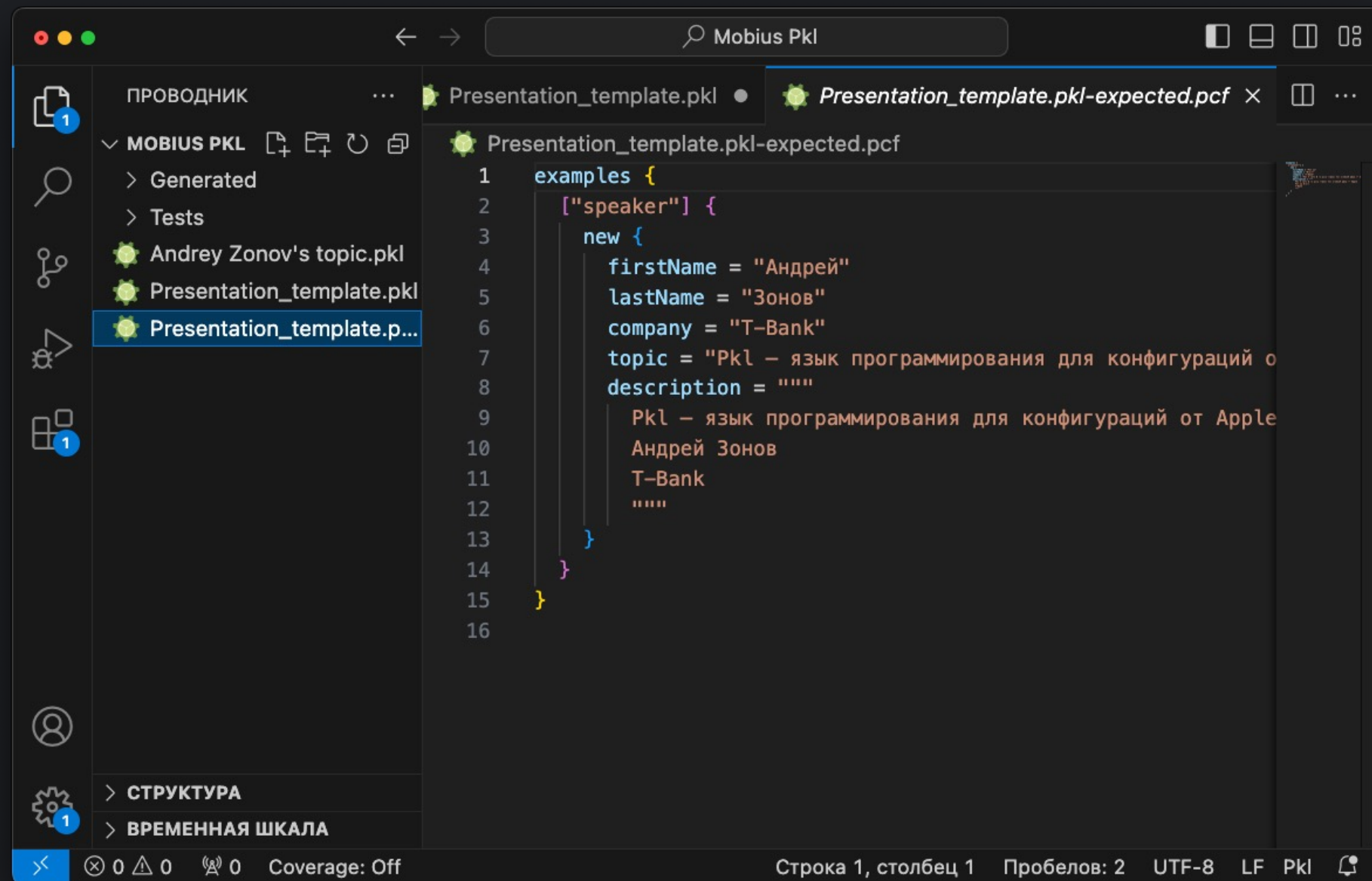


```
Mobius Pkl — azonov@macbook-NQPV414HTG —...
[Mobius Pkl] pkl test Presentation_template.pkl

[module Presentation (file:///Users/azonov/Developer
/Mobius%20Pkl/Presentation_template.pkl, line 1)
  speaker 🖐️
[Mobius Pkl] pkl test Presentation_template.pkl

[module Presentation (file:///Users/azonov/Developer
/Mobius%20Pkl/Presentation_template.pkl, line 1)
  speaker ✅
[Mobius Pkl]
```

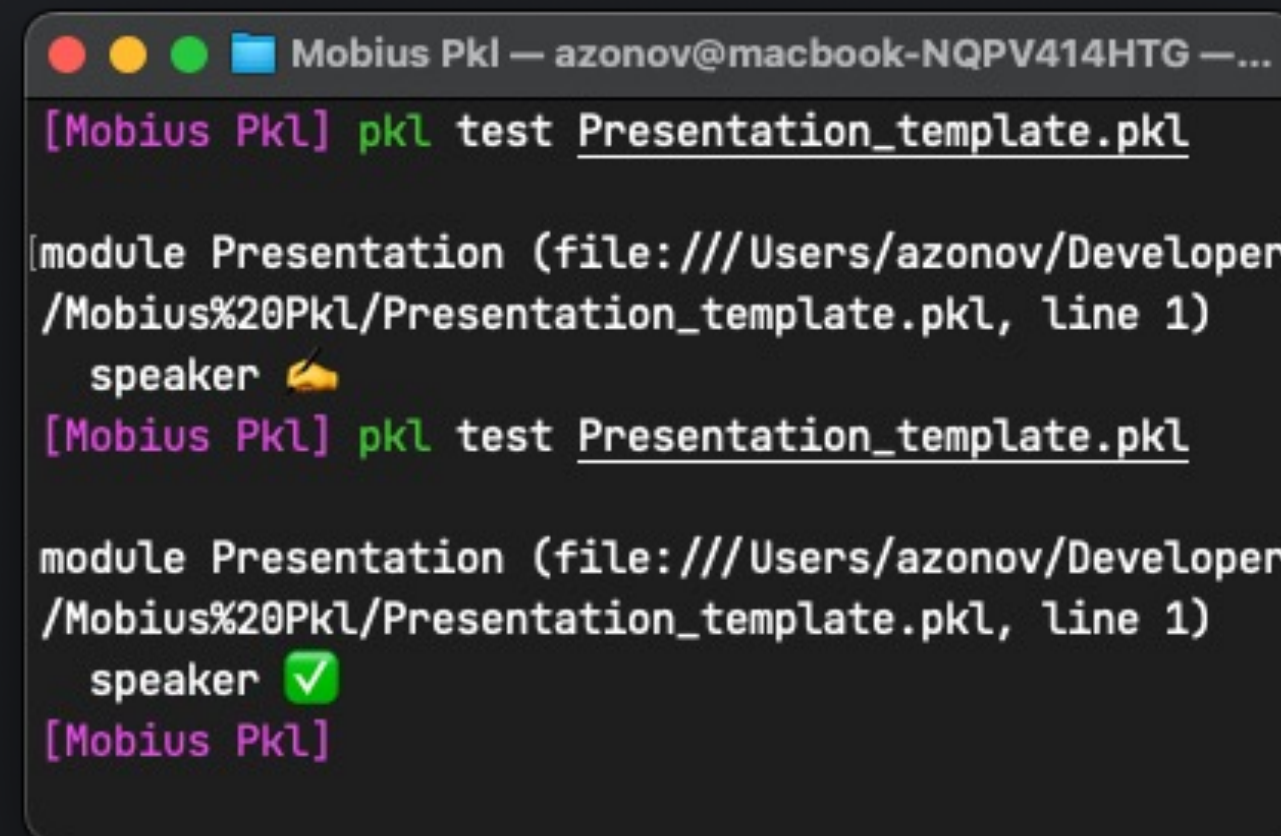
# Тестирование



The screenshot shows an IDE window titled "Mobius PkL". The left sidebar displays a project structure with folders "Generated" and "Tests", and files "Andrey Zonov's topic.pkl", "Presentation\_template.pkl", and "Presentation\_template.p...". The main editor displays the content of "Presentation\_template.pkl-expected.pcf":

```
1 examples {
2   ["speaker"] {
3     new {
4       firstName = "Андрей"
5       lastName = "Зонов"
6       company = "Т-Bank"
7       topic = "PkL - язык программирования для конфигураций о
8       description = ""
9       PkL - язык программирования для конфигураций от Apple
10      Андрей Зонов
11      Т-Bank
12      ""
13    }
14  }
15 }
16
```

- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов
- Example – аналог снимок тестирования для конфига



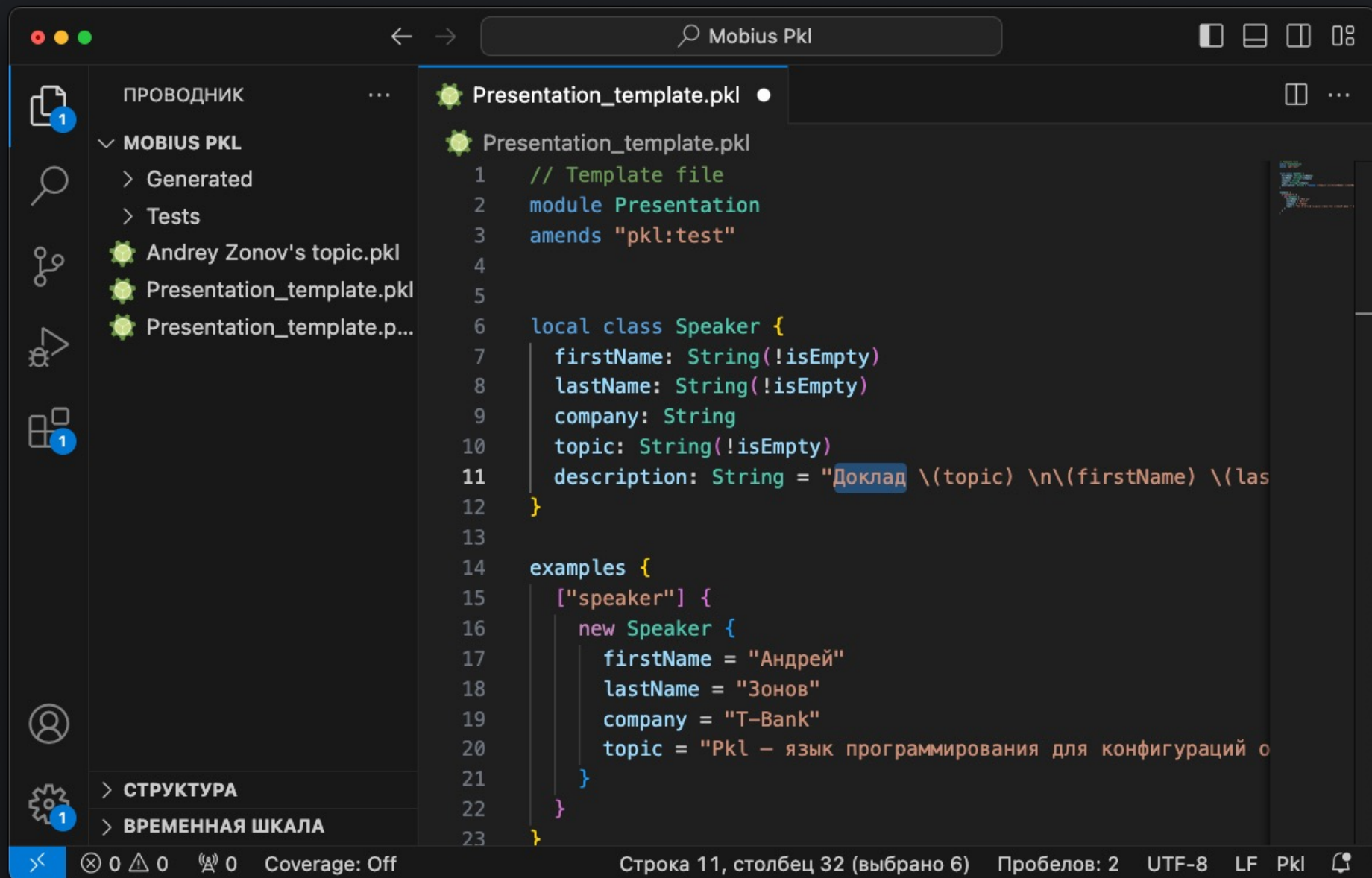
The screenshot shows a terminal window titled "Mobius PkL — azonov@macbook-NQPV414HTG —...". It displays the following output:

```
[Mobius PkL] pkL test Presentation_template.pkl

[module Presentation (file:///Users/azonov/Developer
/Mobius%20PkL/Presentation_template.pkl, line 1)
 speaker 🚫
[Mobius PkL] pkL test Presentation_template.pkl

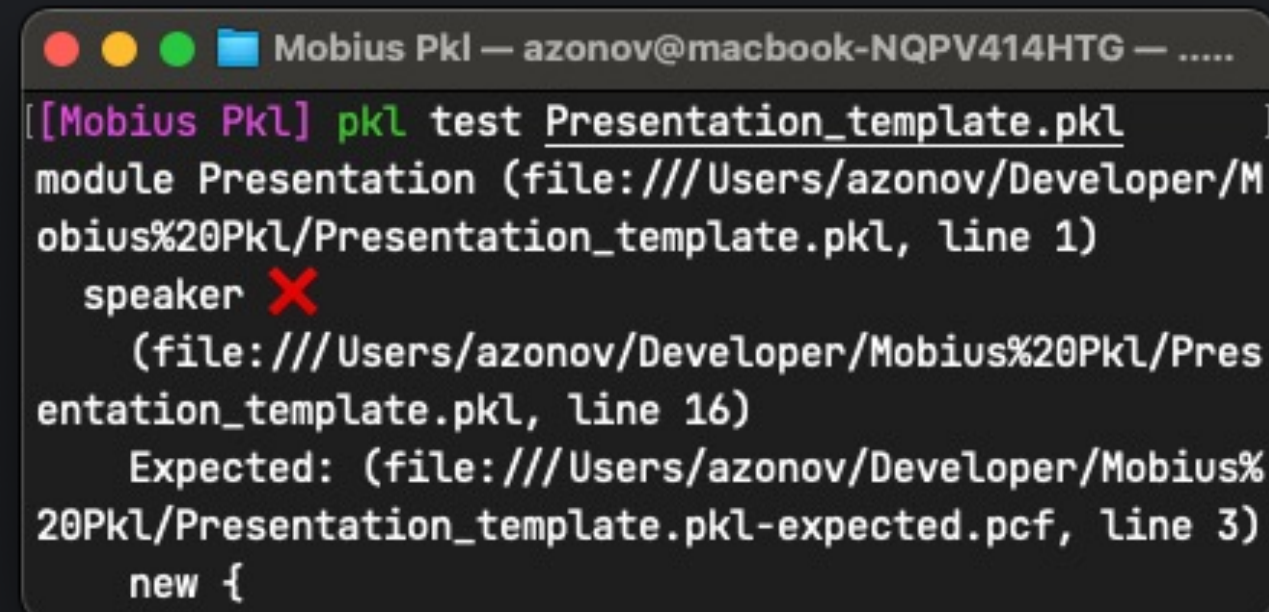
module Presentation (file:///Users/azonov/Developer
/Mobius%20PkL/Presentation_template.pkl, line 1)
 speaker ✅
[Mobius PkL]
```

# Тестирование



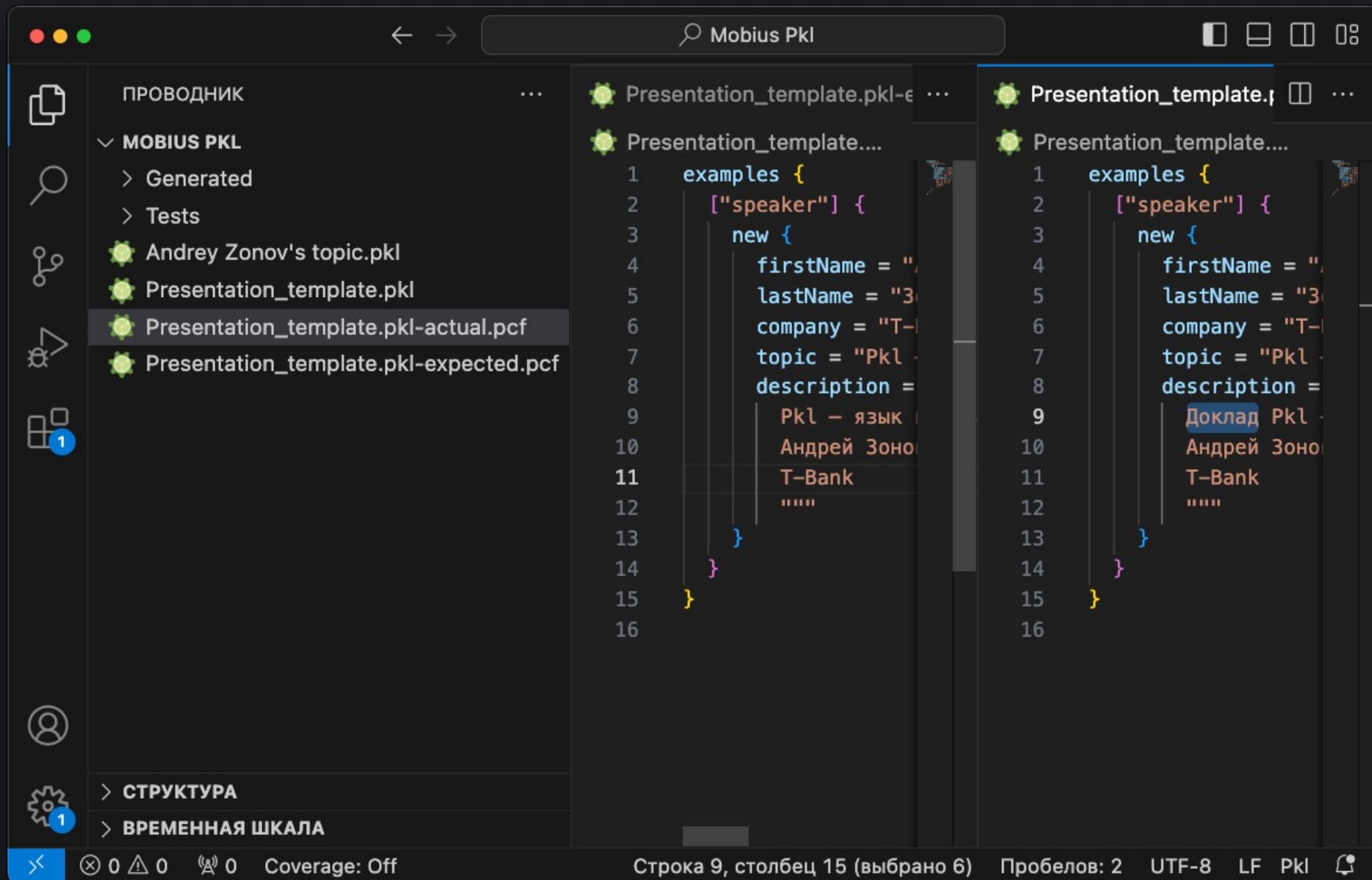
```
1 // Template file
2 module Presentation
3 amends "pkl:test"
4
5
6 local class Speaker {
7   firstName: String(!isEmpty)
8   lastName: String(!isEmpty)
9   company: String
10  topic: String(!isEmpty)
11  description: String = "Доклад \(\topic\) \n\(\firstName\) \(\las
12 }
13
14 examples {
15   ["speaker"] {
16     new Speaker {
17       firstName = "Андрей"
18       lastName = "Зонов"
19       company = "Т-Банк"
20       topic = "Pkl - язык программирования для конфигураций о
21     }
22   }
23 }
```

- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов
- Example – аналог снимот тестирования для конфига



```
Mobius Pkl — azonov@macbook-NQPV414HTG — .....
[[Mobius Pkl] pkl test Presentation_template.pkL
module Presentation (file:///Users/azonov/Developer/M
obius%20Pkl/Presentation_template.pkL, line 1)
speaker X
(file:///Users/azonov/Developer/Mobius%20Pkl/Pres
entation_template.pkL, line 16)
Expected: (file:///Users/azonov/Developer/Mobius%
20Pkl/Presentation_template.pkL-expected.pcf, line 3)
new {
```

# Тестирование



- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов
- Example – аналог снимот тестирования для конфига

```
Mobius PkL — azonov@macbook-NQPV414HTG — .....  
[[Mobius PkL] pkl test Presentation_template.pkl  
module Presentation (file:///Users/azonov/Developer/Mobius%20PkL/Presentation_template.pkl, line 1)  
  speaker X  
    (file:///Users/azonov/Developer/Mobius%20PkL/Presentation_template.pkl, line 16)  
    Expected: (file:///Users/azonov/Developer/Mobius%20PkL/Presentation_template.pkl-expected.pcf, line 3)  
  new {
```

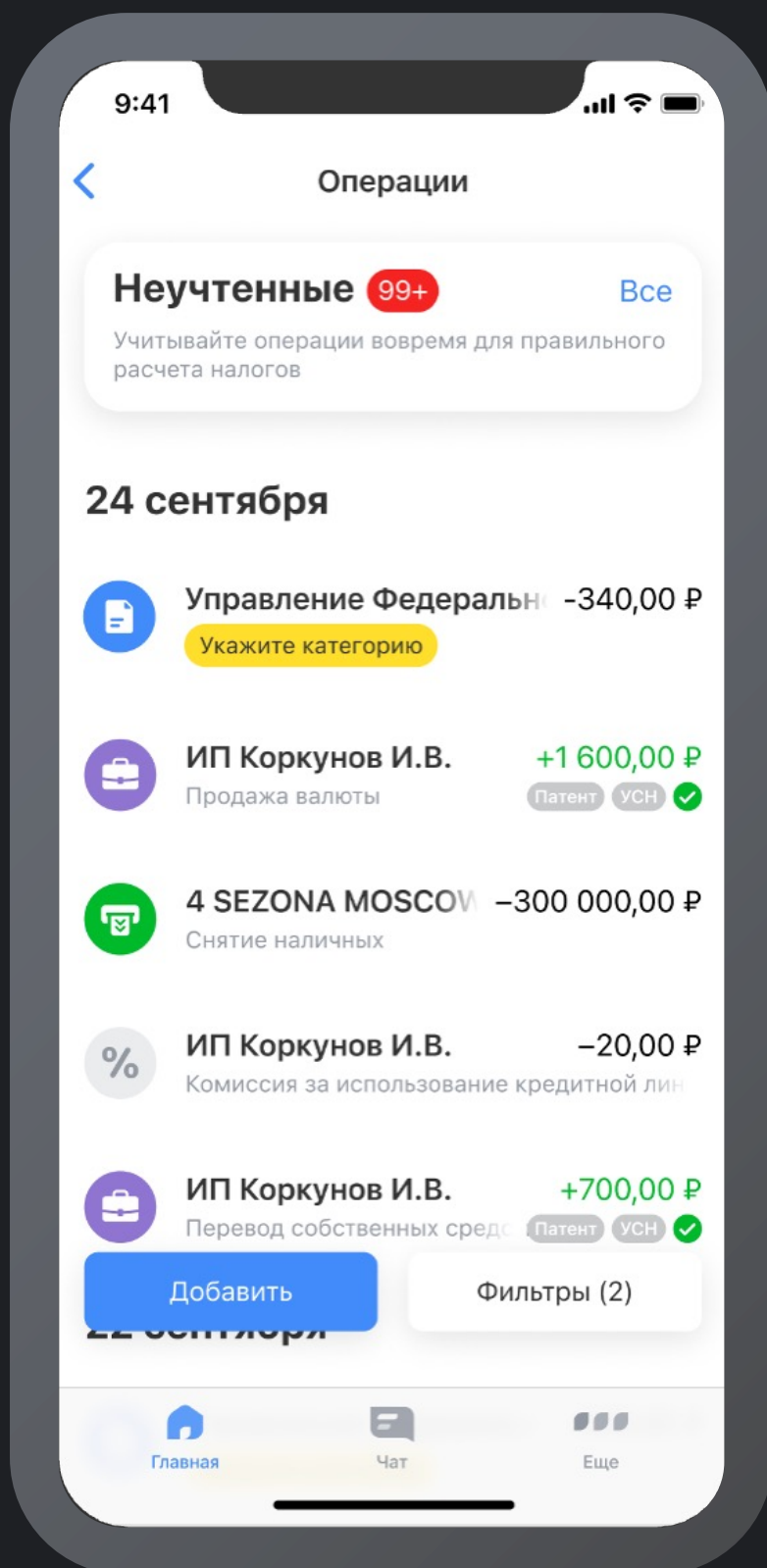
# Тестирование

```
1 examples {
2   ["speaker"] {
3     new {
4       firstName = "Андрей"
5       lastName = "Зонов"
6       company = "Т-Bank"
7       topic = "Pkl - язык программирования для"
8       description = ""
9       Доклад Pkl - язык программирования для
10      Андрей Зонов
11      Т-Bank
12      ""
13    }
14  }
15 }
16
```

- Тесты поддерживаются из коробки
- Fact – обычный логический ассерт, подходит для юнит тестов
- Example – аналог снимот тестирования для конфига

```
Mobius Pkl — azonov@macbook-NQPV414HTG — ../Mobiu...
[[Mobius Pkl] pkl test Presentation_template.pkl --overwrite ]
module Presentation (file:///Users/azonov/Developer/Mobius%2
0Pkl/Presentation_template.pkl, line 1)
  speaker 🍌
[[Mobius Pkl] pkl test Presentation_template.pkl ]
module Presentation (file:///Users/azonov/Developer/Mobius%2
0Pkl/Presentation_template.pkl, line 1)
  speaker ✅
[[Mobius Pkl]
```





# Мокирование

- Пагинация
- Ограничения разных полей
- Уникальность

# Мокирование

The screenshot shows an IDE window titled "Mobius Pkl". The left sidebar shows a file explorer with "MOBIUS PKL" containing "Andrey Zonov's topic.pkl" and "Presentation\_template.pkl". The main editor displays the PKL code for "Andrey Zonov's topic.pkl":

```
1 import "Presentation_template.pkl"
2
3 hidden speaker: Presentation_template.Speaker = new {
4     company = "T-Bank"
5 }
6 speakers {
7     for (offset in List(0, 1, 2)) {
8         (speaker) {
9             name = "Andrey \ \(offset + 1)"
10            yearsOfExperience = 12 + offset
11            email = "an.zonov\ \(offset)@tbank.ru"
12        }
13    }
14 }
```

Below the editor is a terminal window with the command and its output:

```
PROБЛЕМЫ 6 ТЕРМИНАЛ ...
Mobius Pkl > pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "speakers": [
    {
      "name": "Andrey 1",
      "company": "T-Bank",
      "yearsOfExperience": 12,
      "email": "an.zonov0@tbank.ru"
    },
    {
      "name": "Andrey 2",
      "company": "T-Bank",
      "yearsOfExperience": 13,
      "email": "an.zonov1@tbank.ru"
    },
    {
      "name": "Andrey 3",
      "company": "T-Bank",
      "yearsOfExperience": 14,
      "email": "an.zonov2@tbank.ru"
    }
  ]
}
```

The status bar at the bottom shows "Строка 7, столбец 30 Пробелов: 2 UTF-8 LF Pkl".

- Задача написать тест на пагинацию
- В зависимости от параметров, количество отличается
- Часть полей должна быть уникальна

1

# Мокирование

The screenshot shows the Mobius Pkl IDE interface. The top bar displays the project name 'Mobius Pkl'. The left sidebar shows a file explorer with a folder named 'МОБИУС PKL' containing two files: 'Andrey Zonov's topic.pkl' and 'Presentation\_template.pkl'. The main editor window shows the code for 'Andrey Zonov's topic.pkl' with the following content:

```
1 import "Presentation_template.pkl"
2
3 hidden speaker: Presentation_template.Speaker = new {
4     company = "T-Bank"
5 }
6 speakers {
7     for (offset in List(0, 1, 2)) {
8         (speaker) {
9             name = "Andrey \ \(offset + 1)"
10            yearsOfExperience = 12 + offset
11            email = "an.zonov\ \(offset)@tbank.ru"
12        }
13    }
14 }
```

Below the code editor is a terminal window with the following command and output:

```
PROБЛЕМЫ 6 ТЕРМИНАЛ ...
Mobius Pkl > pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "speakers": [
    {
      "name": "Andrey 1",
      "company": "T-Bank",
      "yearsOfExperience": 12,
      "email": "an.zonov0@tbank.ru"
    },
    {
      "name": "Andrey 2",
      "company": "T-Bank",
      "yearsOfExperience": 13,
      "email": "an.zonov1@tbank.ru"
    },
    {
      "name": "Andrey 3",
      "company": "T-Bank",
      "yearsOfExperience": 14,
      "email": "an.zonov2@tbank.ru"
    }
  ]
}
```

The bottom status bar shows 'Строка 7, столбец 30 Пробелов: 2 UTF-8 LF Pkl'.

- Возможность динамически генерировать поддерживаемые МОКИ
- Единая логика для клиентов и для сервера

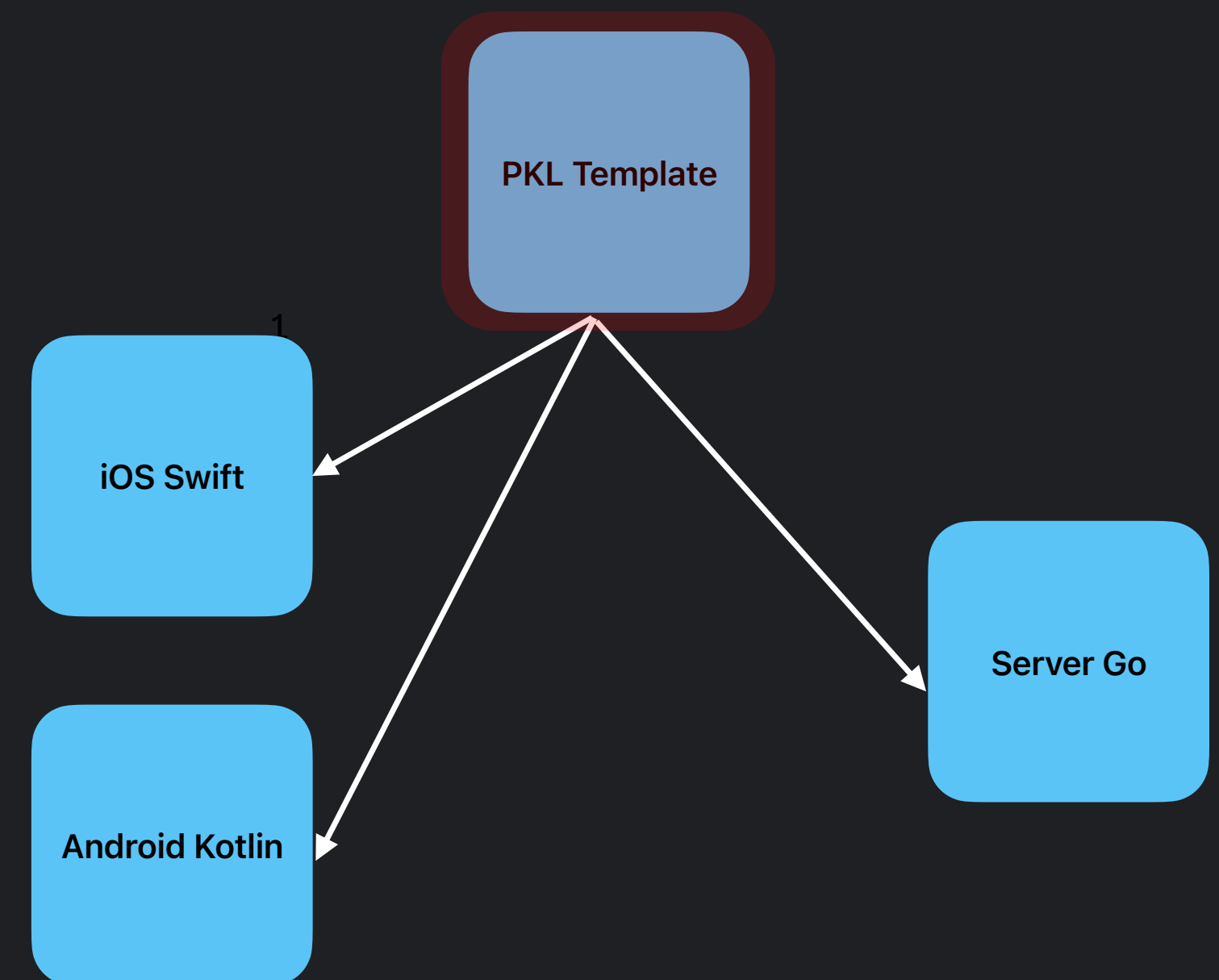
# Мокирование

The screenshot shows an IDE window titled "Mobius Pkl". The editor displays a PKL script in "Andrey Zonov's topic.pkl" that imports "Presentation\_template.pkl" and defines a list of speakers. The terminal shows the command `pkl eval --format json Andrey\ Zonov\'s\ topic.pkl` and its output, a JSON array of speaker objects.

```
1 import "Presentation_template.pkl"
2
3 hidden speaker: Presentation_template.Speaker = new {
4     company = "T-Bank"
5 }
6 speakers {
7     for (offset in List(0, 1, 2)) {
8         (speaker) {
9             name = "Andrey \ \(offset + 1)"
10            yearsOfExperience = 12 + offset
11            email = "an.zonov\ \(offset)@tbank.ru"
12        }
13    }
14 }
```

```
Mobius Pkl > pkl eval --format json Andrey\ Zonov\'s\ topic.pkl
{
  "speakers": [
    {
      "name": "Andrey 1",
      "company": "T-Bank",
      "yearsOfExperience": 12,
      "email": "an.zonov0@tbank.ru"
    },
    {
      "name": "Andrey 2",
      "company": "T-Bank",
      "yearsOfExperience": 13,
      "email": "an.zonov1@tbank.ru"
    },
    {
      "name": "Andrey 3",
      "company": "T-Bank",
      "yearsOfExperience": 14,
      "email": "an.zonov2@tbank.ru"
    }
  ]
}
```

- Возможность динамически генерировать поддерживаемые МОКИ
- Единая логика для клиентов и для сервера



```

/// When the above code is evaluated for the first time, the fo
/// ```
/// examples {
///   ["create Pigeons"] {
///     new {
///       name = "Pigeon"
///       age = 21
///     }
///     new {
///       name = "Pigeon"
///       age = 42
///     }
///   }
/// }
/// ```
///
/// To update expected results after making changes,
/// delete _$filename-expected.pcf_ and evaluate the test modul
examples: Mapping<String, Listing<unknown>>?

```

```

/// The results of running a module's benchmarks.
class BenchmarkReport {
  /// The results of the microbenchmarks that were run.
  ///
  /// Mapping keys are the benchmarks' descriptive names.
  microbenchmarks: Mapping<String, BenchmarkResult>?

  /// The results of the output benchmarks that were run.
  ///
  /// Mapping keys are the benchmarks' descriptive names.
  outputBenchmarks: Mapping<String, BenchmarkResult>?

  /// The results of the parser benchmarks that were run.
  ///
  /// Mapping keys are the benchmarks' descriptive names.
  parserBenchmarks: Mapping<String, BenchmarkResult>?

  /// The platform that benchmarks were run on.
  platform: _platform.Platform
}

```

# PKI

## Standard Library

```

/// A normal version number MUST take the form X.Y.Z
function isNormal(): Boolean = preRelease == null &&

/// Tells if this version has a non-zero [major] and
function isStable(): Boolean = major > 0 && preRelease

/// Strips [preRelease] and [build] from this version
function toNormal(): Version = (this) { preRelease =

function toString() = "\(major).\\(minor).\\(patch)\(if

local function isPreReleaseLessThan(other: Version):
  if (preRelease == null) false
  else if (other.preRelease == null) true
  else if (preRelease == other.preRelease) false
  else

```

▼ stdlib

- 📄 Benchmark.pkl
- 📄 DocPackageInfo.pkl
- 📄 DocsiteInfo.pkl
- 📄 EvaluatorSettings.pkl
- 📄 Project.pkl
- 📄 base.pkl
- 📄 doc-package-info.pkl
- 📄 gradle.lockfile
- 📄 json.pkl
- 📄 jsonnet.pkl
- 📄 math.pkl
- 📄 platform.pkl
- 📄 protobuf.pkl
- 📄 reflect.pkl
- 📄 release.pkl
- 📄 semver.pkl
- 📄 settings.pkl
- 📄 shell.pkl
- 📄 stdlib.gradle.kts
- 📄 test.pkl
- 📄 xml.pkl
- 📄 yaml.pkl

```

/// When the above code is evaluated for the first time, the fo
/// ```
/// examples {
///     ["create Pigeons"] {
///         new {
///             name = "Pigeon"
///             age = 21
///         }
///         new {
///             name = "Pigeon"
///             age = 42
///         }
///     }
/// }
/// ```
///
/// To update expected results after making changes,
/// delete _$filename-expected.pcf_ and evaluate the test modul
examples: Mapping<String, Listing<unknown>>?

```

```

/// The results of running a module's benchmarks.
class BenchmarkReport {
    /// The results of the microbenchmarks that were run.
    ///
    /// Mapping keys are the benchmarks' descriptive names.
    microbenchmarks: Mapping<String, BenchmarkResult>?

    /// The results of the output benchmarks that were run.
    ///
    /// Mapping keys are the benchmarks' descriptive names.
    outputBenchmarks: Mapping<String, BenchmarkResult>?

    /// The results of the parser benchmarks that were run.
    ///
    /// Mapping keys are the benchmarks' descriptive names.
    parserBenchmarks: Mapping<String, BenchmarkResult>?

    /// The platform that benchmarks were run on.
    platform: _platform.Platform
}

```

# PKI

## Standard Library

```

/// A normal version number MUST take the form X.Y.Z
function isNormal(): Boolean = preRelease == null &&

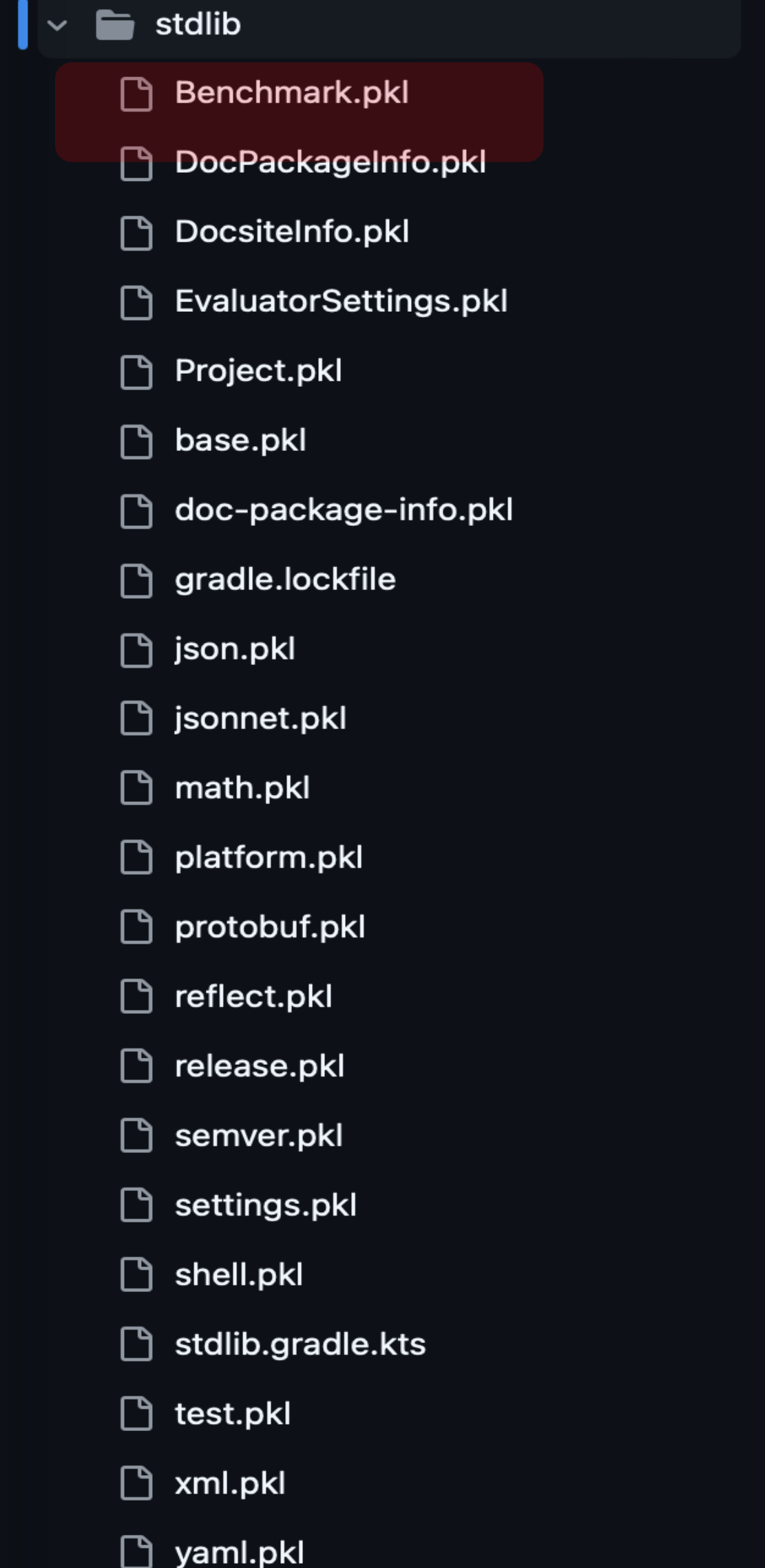
/// Tells if this version has a non-zero [major] and
function isStable(): Boolean = major > 0 && preRelease

/// Strips [preRelease] and [build] from this version
function toNormal(): Version = (this) { preRelease =

function toString() = "\(major).\\(minor).\\(patch)\(if

local function isPreReleaseLessThan(other: Version):
    if (preRelease == null) false
    else if (other.preRelease == null) true
    else if (preRelease == other.preRelease) false
    else

```



```
/// When the above code is evaluated for the first time, the fo
/// ```
/// examples {
///   ["create Pigeons"] {
///     new {
///       name = "Pigeon"
///       age = 21
///     }
///     new {
///       name = "Pigeon"
///       age = 42
///     }
///   }
/// }
/// ```
///
/// To update expected results after making changes,
/// delete _$filename-expected.pcf_ and evaluate the test modul
examples: Mapping<String, Listing<unknown>>?
```

```
/// The results of running a module's benchmarks.
class BenchmarkReport {
  /// The results of the microbenchmarks that were run.
  ///
  /// Mapping keys are the benchmarks' descriptive names.
  microbenchmarks: Mapping<String, BenchmarkResult>?

  /// The results of the output benchmarks that were run.
  ///
  /// Mapping keys are the benchmarks' descriptive names.
  outputBenchmarks: Mapping<String, BenchmarkResult>?

  /// The results of the parser benchmarks that were run.
  ///
  /// Mapping keys are the benchmarks' descriptive names.
  parserBenchmarks: Mapping<String, BenchmarkResult>?

  /// The platform that benchmarks were run on.
  platform: _platform.Platform
}
```

# PKI

## Standard Library

```
/// A normal version number MUST take the form X.Y.Z
function isNormal(): Boolean = preRelease == null &&

/// Tells if this version has a non-zero [major] and
function isStable(): Boolean = major > 0 && preRelease

/// Strips [preRelease] and [build] from this version
function toNormal(): Version = (this) { preRelease =

function toString() = "\(major).\\(minor).\\(patch)\(if

local function isPreReleaseLessThan(other: Version):
  if (preRelease == null) false
  else if (other.preRelease == null) true
  else if (preRelease == other.preRelease) false
  else
```

stdlib

- Benchmark.pkl
- DocPackageInfo.pkl
- DocsiteInfo.pkl
- EvaluatorSettings.pkl
- Project.pkl
- base.pkl
- doc-package-info.pkl
- gradle.lockfile
- json.pkl
- jsonnet.pkl
- math.pkl
- platform.pkl
- protobuf.pkl
- reflect.pkl
- release.pkl
- semver.pkl
- settings.pkl
- shell.pkl
- stdlib.gradle.kts
- test.pkl
- xml.pkl
- yaml.pkl

## Basic Pkl/K8s Example

This example demonstrates how to

- convert YAML manifests to Pkl
- evaluate Pkl manifests to YAML
- catch manifest errors during evaluation
- apply Pkl manifests to Kubernetes.

To convert the YAML manifests located in the `yaml` directory to Pkl, run [io.k8s.convert](#) for each

```
$ pkl eval -p input=../../yaml/frontend.yaml \
  -o frontend.pkl \
  package://pkg.pkl-lang.org/pkl-pantry/k8s.contrib@1.0.1#/convert.pkl

$ pkl eval -p input=../../yaml/redis-primary.yaml \
  -o redis-primary.pkl \
  package://pkg.pkl-lang.org/pkl-pantry/k8s.contrib@1.0.1#/convert.pkl

$ pkl eval -p input=../../yaml/redis-secondary.yaml \
  -o redis-secondary.pkl \
  package://pkg.pkl-lang.org/pkl-pantry/k8s.contrib@1.0.1#/convert.pkl
```

## Integration with Go

Pkl provides a rich integration with Go. Our integration allows you to generate code from Pkl source code.

To get started, reference the [Quickstart guide](#). Alternatively, use our [example](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/myteam/myapp/myconfig"
)

func main() {
    cfg, err := myconfig.LoadFromPath(context.Background(), "myconfig.yaml")
    if err != nil {
        panic(err)
    }
}
```

# Pkl

## Пакеты

Для любой задачи

### 4. Evaluate Pkl configuration data in Swift

With the above scaffolding set up, evaluate Pkl configuration data in Swift.

First, define some configuration that uses the Pkl schema.

In our example, we create a file at path `pkl/local/appConfig.pkl`. We imagine that we are defining a local environment, and would likewise place other environments in sibling directories; e.g. `pkl/prod/appConfig.pkl`.

`pkl/local/appConfig.pkl`

```
amends "../AppConfig.pkl"

host = "localhost"

port = 5939
```

Once defined, evaluate the Pkl module into Swift data.

`Sources/MyApplication/Main.swift`

```
func main() async throws {
    let config = try await AppConfig.loadFrom(source: .path("pkl/local/appConfig.pkl"))
    print("I'm running on host \(config.host)\n")
}
```

- packages
  - > com.circleci.v2
  - > com.influxdata.telegraf
  - > io.prometheus
  - > k8s.contrib.appEnvCluster
  - > k8s.contrib.crd
  - > k8s.contrib
  - > org.apache.spark
  - > org.json\_schema.contrib
  - > org.json\_schema
  - > org.openapis.v3.contrib
  - > org.openapis.v3
  - > pkl.csv
  - > pkl.experimental.deepToTyped
  - > pkl.experimental.net
  - > pkl.experimental.syntax
  - > pkl.experimental.uri



## Basic Pkl/K8s Example

This example demonstrates how to

- convert YAML manifests to Pkl
- evaluate Pkl manifests to YAML
- catch manifest errors during evaluation
- apply Pkl manifests to Kubernetes.

To convert the YAML manifests located in the `yaml` directory to Pkl, run [io.k8s.convert](#) for each

```
$ pkl eval -p input=../../yaml/frontend.yaml \
  -o frontend.pkl \
  package://pkg.pkl-lang.org/pkl-pantry/k8s.contrib@1.0.1#/convert.pkl

$ pkl eval -p input=../../yaml/redis-primary.yaml \
  -o redis-primary.pkl \
  package://pkg.pkl-lang.org/pkl-pantry/k8s.contrib@1.0.1#/convert.pkl

$ pkl eval -p input=../../yaml/redis-secondary.yaml \
  -o redis-secondary.pkl \
  package://pkg.pkl-lang.org/pkl-pantry/k8s.contrib@1.0.1#/convert.pkl
```

## Integration with Go

Pkl provides a rich integration with Go. Our integration allows you to generate code from Pkl source code.

To get started, reference the [Quickstart guide](#). Alternatively, use our [example](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/myteam/myapp/myconfig"
)

func main() {
    cfg, err := myconfig.LoadFromPath(context.Background(), "myconfig.yaml")
    if err != nil {
        panic(err)
    }
}
```

# Pkl

## Пакеты

Для любой задачи

### 4. Evaluate Pkl configuration data in Swift

With the above scaffolding set up, evaluate Pkl configuration data in Swift.

First, define some configuration that uses the Pkl schema.

In our example, we create a file at path `pkl/local/appConfig.pkl`. We imagine that we are defining a local environment, and would likewise place other environments in sibling directories; e.g. `pkl/prod/appConfig.pkl`.

`pkl/local/appConfig.pkl`

```
amends "../AppConfig.pkl"

host = "localhost"

port = 5939
```

Once defined, evaluate the Pkl module into Swift data.

`Sources/MyApplication/Main.swift`

```
func main() async throws {
    let config = try await AppConfig.loadFrom(source: .path("pkl/local/appConfig.pkl"))
    print("I'm running on host \(config.host)\n")
}
```

- packages
  - > com.circleci.v2
  - > com.influxdata.telegraf
  - > io.prometheus
  - > k8s.contrib.appEnvCluster
  - > k8s.contrib.crd
  - > k8s.contrib
  - > org.apache.spark
  - > org.json\_schema.contrib
  - > org.json\_schema
  - > org.openapis.v3.contrib
  - > org.openapis.v3
  - > pkl.csv
  - > pkl.experimental.deepToTyped
  - > pkl.experimental.net
  - > pkl.experimental.syntax
  - > pkl.experimental.uri

# Package OpenAPI

The screenshot shows an IDE with two tabs: `petstore.pkl` and `petstore.yaml`. The left pane displays a JSON object representing an OpenAPI path, and the right pane displays the equivalent YAML representation.

```
28 paths {
29   ["/pets"] {
30     get {
31       summary = "List all pets"
32       operationId = "listPets"
33       tags {
34         "pets"
35       }
36       parameters {
37         new {
38           name = "limit"
39           `in` = "query"
40           description = "How many items to return at one time (max 100)"
41           required = false
42           schema {
43             type = "integer"
44             maximum = 100
45             format = "int32"
46           }
47         }
48       }
49       responses {
50         ["200"] {
51           description = "A paged array of pets"
52           headers {
53             ["x-next"] {
54               description = "A link to the next page of responses"
55               schema {
56                 type = "string"
57               }
58             }
59           }
60           content {
61             ["application/json"] {
```

```
9 paths:
10 /pets:
11 get:
12   summary: List all pets
13   operationId: listPets
14   tags:
15     - pets
16   parameters:
17     - name: limit
18       in: query
19       description: How many items to return at one time (max 100)
20       required: false
21       schema:
22         type: integer
23         maximum: 100
24         format: int32
25   responses:
26     '200':
27       description: A paged array of pets
28       headers:
29         x-next:
30           description: A link to the next page of responses
31           schema:
32             type: string
33       content:
34         application/json:
35           schema:
36             $ref: "#/components/schemas/Pets"
37       default:
38         description: unexpected error
39         content:
40           application/json:
41             schema:
42               $ref: "#/components/schemas/Error"
```

Строка 1, столбец 1 Пробелов: 2 UTF-8 LF YAML

# Выводы



## Язык для работы с конфигурациями

JSON/YAML контракты



## Не является заменой OpenAPI

Ограничивается моделями



## Open source

Позволяет интегрировать без «вендерлока»



## Можно начать использовать

Подходит для зрелого проекта



# Андрей Зонов

## Stuff iOS | Руководитель группы

 [linkedin.com/in/avzonov](https://www.linkedin.com/in/avzonov)

 [t.me/avzonov](https://t.me/avzonov)





**Спасибо!**