

## ИНДИВИДУАЛЬНЫЙ ПЛАН (ЗАДАНИЕ И ГРАФИК) ПРОВЕДЕНИЯ ПРАКТИКИ

Сопрачев Андрей Константинович

**Направление подготовки (код/наименование):** 09.04.04 «Программная инженерия»

**Профиль (код/наименование):** 09.04.04\_01 «Технология разработки и сопровождения качественного программного продукта»

**Вид практики:** научно-исследовательская работа

**Тип практики:** концентрированная

**Место прохождения практики:** ФГАОУ ВО «СПбПУ», ИКНК, ВШПИ, СПб, ул. Политехническая, 29

**Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»:**

Дробинцев Павел Дмитриевич, к.т.н, доцент ВШПИ ИКНК


**Руководитель практической подготовки от профильной организации: -**

**Рабочий график проведения практики**

**Сроки практики:** с 03.02.25 по 31.03.25

№ п/п	Этапы (периоды) практики	Вид работ	Сроки прохождения этапа (периода) практики
1	Организационный этап	Установочная лекция для разъяснения целей, задач, содержания и порядка прохождения практики, инструктаж по технике безопасности, выдача сопроводительных документов по практике	30.01
2	Основной этап	Разработка и архитектурные особенности PolyMap: платформа для проектирования и размещения интерактивных карт помещений и территорий. Применение Serverless-подхода.	30.01 – 01.04
3	Заключительный этап	Защита отчета по практике	02.04

Обучающийся

 / Сопрачев А. К. /

Руководитель практической подготовки  
от ФГАОУ ВО «СПбПУ»

 / Дробинцев П. Д. /

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»  
Институт компьютерных наук и кибербезопасности  
Высшая школа программной инженерии

**Отчет о прохождении научно-исследовательской работы**

Сопрачев Андрей Константинович

2 курс, 5140904/20102

09.04.04 «Программная инженерия»

**Место прохождения практики:** ФГАОУ ВО «СПбПУ», ИКНК, ВШПИ, СПб, ул. Политехни-  
ческая, 29

**Сроки практики:** с 03.02.25 по 31.03.25

**Руководитель практической подготовки от ФГАОУ ВО «СПбПУ»:**

Дробинцев Павел Дмитриевич, к.т.н, доцент ВШПИ ИКНК

**Руководитель практической подготовки от профильной организации: -**

**Оценка: зачтено**

Руководитель практической подготовки  
от ФГАОУ ВО «СПбПУ»



/ Дробинцев П. Д. /

Руководитель практической подготовки  
от профильной организации

/ - /

Обучающийся



/ Сопрачев А. К. /

Дата: 31.03.25

## СОДЕРЖАНИЕ

1. Постановка задачи .....	4
2. Актуальность работы .....	4
2.1. Статистика PolyMap .....	4
2.2. Задачи для развития приложения PolyMap .....	6
3. Подходы к решению задачи .....	6
3.1. Serverless подход .....	7
3.2. Общая архитектура приложения .....	8
3.3. Детальная архитектура приложения .....	9
4. CI/CD .....	10
4.1. Terraform .....	10
4.2. Пайплайн в GitHub Actions .....	12
Список литературы .....	17

## **1. ПОСТАНОВКА ЗАДАЧИ**

Задачей проекта является продолжить развитие приложение PolyMap[1]. Необходимо разработать универсальный вариант приложения, позволяющий отображать различные карты разных заказчиков. Так же приложение должно работать в браузерном режиме и предоставляться в виде SaaS[2] (Software as a Service).

## **2. АКТУАЛЬНОСТЬ РАБОТЫ**

Актуальность работы обусловлена тем, что существующие решения не позволяют отображать карты разных заказчиков в одном приложении. Так же существующие решения не позволяют отображать карты в браузерном режиме. Проблема навигации в пешеходной среде является актуальной для многих публичных мест. Приложение PolyMap крайне хорошо справляется с этой задачей, и обладает высокими статистическими показателями.

### **2.1. Статистика PolyMap**

Согласно Yandex App Metrica[3], еженедельное число пользователей приложения PolyMap (Рис. 1) составляет 4-5 тысяч уникальных человек. При этом пик поиска аннотаций приходится на начало сентября: 3 тысячи пользователей (Рис. 2) просматривали аннотации 27 тысяч раз (Рис. 3) в неделю с 4 по 10 сентября.

Это говорит о том, что приложение PolyMap пользуется спросом у пользователей, и имеет потенциал для развития.

## Аудитория

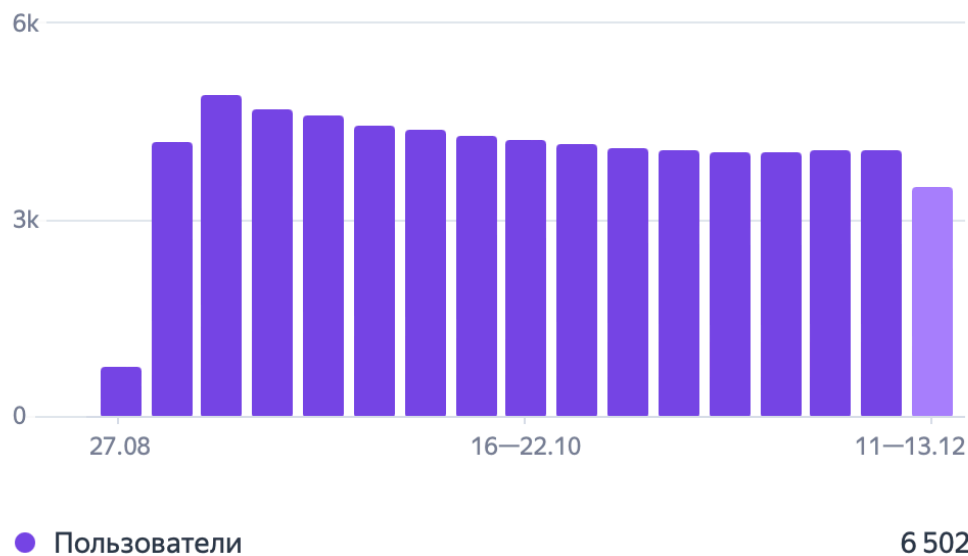


Рис. 1. Ежедневное количество пользователей PolyMap

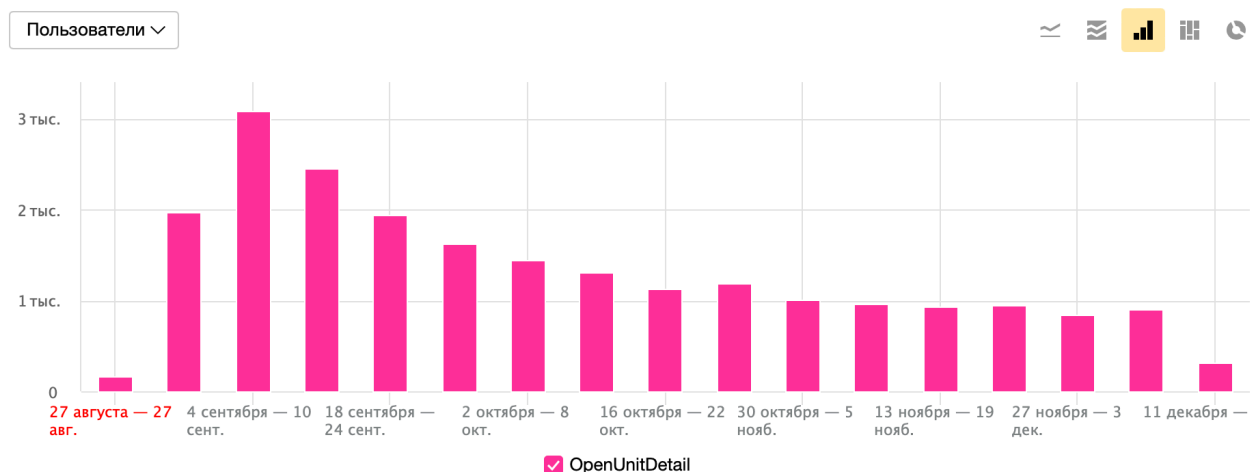


Рис. 2. Ежедневное количество пользователей открывших информацию об аннотации

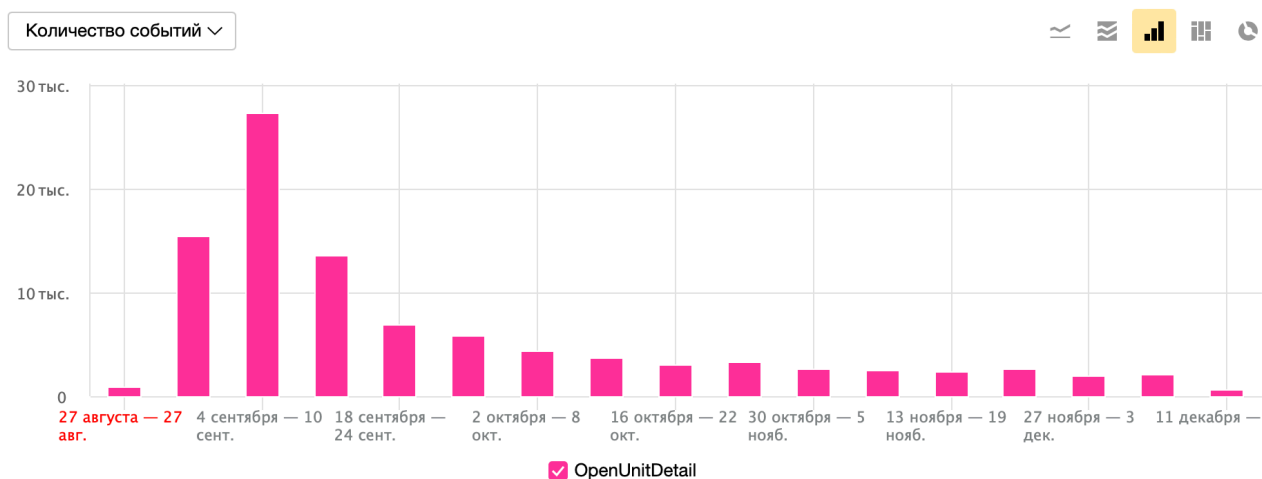


Рис. 3. Ежедневное число открытий аннотаций

## **2.2. Задачи для развития приложения PolyMap**

Основной проблемой приложения PolyMap является то, что оно не универсально. Приложение PolyMap разработано для конкретного вуза, и не может быть использовано другими. Кроме того, приложение разработано только для операционной системы iOS, и не может быть использовано на других платформах.

Из этих проблем вытекают следующие задачи для развития приложения PolyMap:

1. Разработать универсальный вариант приложения, позволяющий отображать различные карты разных заказчиков.
2. Разработать удобный конструктор карт, позволяющий заказчикам создавать и обновлять карты самостоятельно.
3. Разработать веб-приложение способное отображать карту в браузере на любой платформе.

При разработке так же важно предусмотреть возможность масштабирования приложения под большие нагрузки.

## **3. ПОДХОДЫ К РЕШЕНИЮ ЗАДАЧИ**

Несмотря на кажущуюся простоту задачи, приложение PolyMap состоит из большого числа модулей, выполняющих разные задачи. Сейчас все серверные части PolyMap написаны с использованием монолитного подхода и развернуты на виртуальной машине. Это позволяет быстро разрабатывать и развивать приложение, но не позволяет масштабировать его под большие нагрузки.

Для решения задачи развития приложения PolyMap необходимо переписать серверную часть приложения с использованием микросервисной архитектуры. Это позволит разрабатывать и развивать приложение в дальнейшем, а так же масштабировать его под большие нагрузки.

При использовании микросервисного подхода возникает ряд сложностей с оркестрацией и развертыванием приложения. Для решения этих проблем необходимо использовать контейнеризацию Docker[4]. Для оркестрации контейнеров наиболее распространённым подходом является использование Kubernetes[5], однако этот подход требует больших затрат на поддержку, а так же требует

написания специфичного кода. Для решения этих проблем можно использовать новый Serverless подход, который позволяет автоматизировать развертывание и поддержку приложения.

### 3.1. Serverless подход

Serverless[6] - это модель облачных вычислений, при которой облачный провайдер полностью управляет запуском и инфраструктурой сервера. Пользователи пишут только код приложения. Основными преимуществами Serverless подхода являются:

- Автомасштабирование. Сервис автоматически масштабируется в зависимости от нагрузки, без необходимости управления инфраструктурой
- Оплата по использованию (Pay-as-you-go). Пользователь платит только за реально использованные ресурсы
- Администрирование. Пользователь не заботится о настройке и обслуживании серверов, администрирование серверов полностью ложится на облачного провайдера. Снижает инфраструктурные риски и упрощает разработку приложения
- Быстрое развертывание. Приложение разворачивается в несколько кликов, без необходимости настройки серверов

#### Yandex Cloud

В качестве облачного провайдера для развёртывания приложения PolyMap была выбрана Yandex Cloud[7]. Yandex Cloud предоставляет полный набор инструментов для разработки и развёртывания Serverless приложений. Основными используемыми инструментами являются:

- Serverless Containers[8]. Позволяет разворачивать Docker контейнеры в облаке, и автоматически масштабировать их в зависимости от нагрузки
- API Gateway[9]. Позволяет создавать API для взаимодействия с приложением
- Yandex Object Storage[10]. Облачное Serverless хранилище данных, используется для хранения статических файлов приложения по протоколу S3[11]
- Yandex Database[12]. Облачная Serverless база данных, используется для хранения данных приложения по протоколу DynamoDB

Все эти инструменты позволяют разворачивать приложение в облаке, и автоматически масштабировать его в зависимости от нагрузки. При этом пользователь платит только за реально использованные ресурсы. Потенциал горизонтального масштабирования неограничен, и позволяет разворачивать приложение на любое количество серверов. API Gateway балансирует нагрузку между Serverless контейнерами, и при стандартном подходе на каждый запрос создаётся новый контейнер.

### 3.2. Общая архитектура приложения

При разработке высоконагруженных приложений необходимо использовать CDN (Content Delivery Network[13]). CDN - это распределённая сеть серверов, которая позволяет ускорить доставку контента до конечного пользователя. Приложение PolyMap использует карты, которые имеют большой объём данных, и для ускорения их доставки необходимо использовать CDN.

В качестве CDN необходимо использовать провайдера с большим покрытием серверов, к сожалению у Yandex Cloud нет CDN. Поэтому в качестве CDN был выбран Cloudflare[14]. Cloudflare имеет большое количество серверов по всему миру, и позволяет ускорить доставку контента до конечного пользователя.

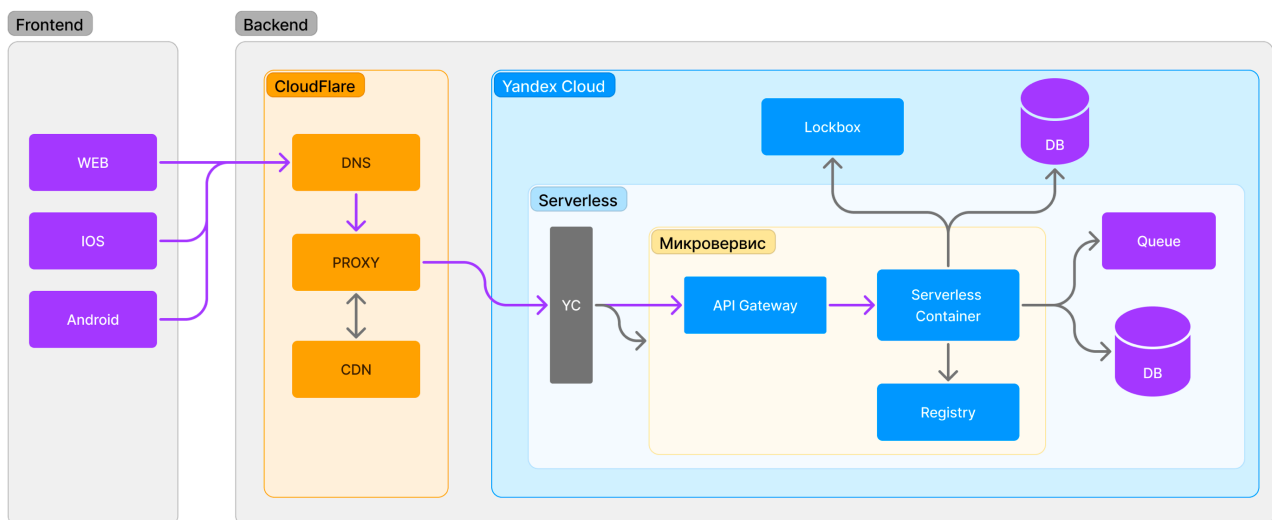


Рис. 4. Общая архитектура приложения PolyMap

Когда фронтенд совершает запрос, он попадает на Cloudflare DNS, который определяет на какой сервер отправить запрос. В случае если запрос разрешает кэширование, то Cloudflare возвращает ответ из кеша без обращения к серверу. Если запрос не разрешает кэширование, то Cloudflare проксирует запрос на



API Gateway в Яндексе. API Gateway балансирует нагрузку между Serverless контейнерами, образы которых хранятся в Container Registry. Контейнеры могут получать доступ к различным репозиториям данных, расположенным как в Serverless окружение, так и просто в облаке.

### **3.3. Детальная архитектура приложения**

Приложение состоит из следующих микросервисов:

1. Конструктор карты
  - 1.а. `constructor-ws`: WebSocket[15] сервер, обеспечивает синхронизацию данных между пользователями при работе с одной картой
  - 1.б. `blobl-storage`: хранилище данных, используется для хранения больших пользоательских файлов при создании карты (например планировки помещений)
  - 1.в. `constructor-back`: бекенд, обеспечивает работу конструктора карт. Права доступа, авторизацию, сохранение и т.д.
2. `map-storage`: отвечает за хранение и раздачу карт
3. `share-back`: отвечает за генерацию коротких ссылок для приглашения пользователей в карту/маршрут/аннотацию
4. `qr-generator`: отвечает за генерацию QR кодов для приглашения
5. Поддержка веб-версии карты
  - 5.а. `web-map-front`: раздаёт статические файлы для веб-версии карты
  - 5.б. `web-map-back`:: динамически генерирует `index.html` для веб-версии карты, для корректной работы SEO оптимизации



создавать полную инфраструктуру в облаке с использованием высокоуровневого конфигурационного языка. Это означает, что описание инфраструктуры хранится в виде кода в репозитории, и может быть использовано для развёртывания и управления инфраструктурой.

Основными преимуществами Terraform являются:

- IaC: Terraform позволяет определять инфраструктуру с помощью кода, что обеспечивает более высокий уровень автоматизации, удобство отслеживания изменений, лучшее управление версиями и повторное использование кода
- Автоматизация Развертывания: С Terraform можно автоматически развертывать и обновлять серверные функции, API-интерфейсы и другие ресурсы, необходимые для serverless приложений.
- Модульность: Terraform позволяет создавать переиспользуемые модули
- Планирование Изменений: Terraform предоставляет детализированный план изменений перед их применением, что позволяет предвидеть и управлять последствиями изменений в инфраструктуре

```
resource "docker_image" "main" {
  name = "cr.yandex/${yandex_container_registry.registry.id}/main:latest"
  build { context = abspath(local.app_path) }
  triggers = { hash = local.project_files_hash }
}

resource "yandex_serverless_container" "container" {
  name          = local.project_name
  memory        = 128
  cores         = 1
  core_fraction = 5
  concurrency   = 16
  folder_id     = var.folder_id
  service_account_id = yandex_iam_service_account.registry_puller.id

  image {
    url      = docker_registry_image.registry.name
    digest   = docker_registry_image.registry.sha256_digest
    environment = merge(var.container_env, { "LOG_VARIANT" = "JSON" })
  }

  depends_on = [ ...
  ]
}
```

Рис. 6. Пример использования Terraform

## 4.2. Пайплайн в GitHub Actions

Наиболее популярные подходы к использованию Git[18] подразумевают создание новых веток на каждую новую функциональность, и в момент завершения разработки функциональности создание Pull Request'а в основную ветку. После этого Pull Request проходит ряд проверок, и в случае успешного прохождения проверок, изменения из ветки сливаются в основную ветку.

Репозиторий расположен в GitHub, и используется GitHub Actions[19] для автоматизации процесса развёртывания приложения. GitHub Actions позволяет запускать различные действия при событиях в репозитории, таких как создание Pull Request'а, коммит в ветку и т.д.

Я использую подход GitFlow[20], согласно которому создаются две основные ветки – main и dev, для текущей актуальной версии приложения и для разработки соответственно. Все новые функциональности разрабатываются в отдельных ветках, которые вливаются в ветку dev. После этого ветка dev вливается в ветку main, и выпускается новая версия приложения.

При таком подходе CI/CD система должна обрабатывать:

- Открытие Pull Request'а в ветку dev
  - запуск статического анализатора и тестов
  - проверка terraform конфигурации перед применением
- Коммит в ветку dev (принятие изменений в Pull Request'е автоматически создаёт коммит в ветку dev): сборка и развёртывание изменений в тестовом окружении
- Коммит в ветку main (принятие изменений в Pull Request'е автоматически создаёт коммит в ветку main): сборка и развёртывание изменений в продакшен окружении

Пайплайн для открытия Pull Request'а в ветку dev

Этот пайплайн срабатывает при открытии и изменении Pull Request'а. Благодаря использованию Serverless подхода, появляется возможность создавать НОВОЕ полностью изолированное окружение для каждого PullRequest'а. Это позволяет выполнять тестирование на реальной инфраструктуре при этом не конфликтуя с другими PullRequest'ами. Благодаря использованию Terraform,

временная тестовое окружение будет полностью идентично продакшену, что позволит выявить большинство ошибок на ранних этапах разработки.

Состоит двух параллельных задач:

1. Сборка для тестового окружения:

- 1.а. Lint and Unit Tests: запускается линтер и юнит тесты для всех модулей приложения
- 1.б. Request Worker Folder: запрашивает новую директорию в Yandex Cloud для развёртывания тестового окружения
- 1.в. Clear comments: очищает старый автоматический комментарий в Pull Request'e
- 1.г. Terraform: выполняет развёртывание тестового окружения с помощью Terraform в указанной директории
- 1.д. E2E tests: запускает E2E тесты, передавая в качестве параметра URL до тестового окружения (генерируется автоматически Yandex Cloud)
- 1.е. Comment Deploy Link: создаёт комментарий в Pull Request'e с ссылкой на тестовое окружение для ручного тестирования

2. Подготовка к принятию изменений:

- 2.а. Clear comments: очищает старый автоматический комментарий в Pull Request'e
- 2.б. Terraform: выполняет команду `terraform plan` в продакшен режиме для подготовки к принятию изменений. Вывод команды печатается в комментарий в Pull Request'e

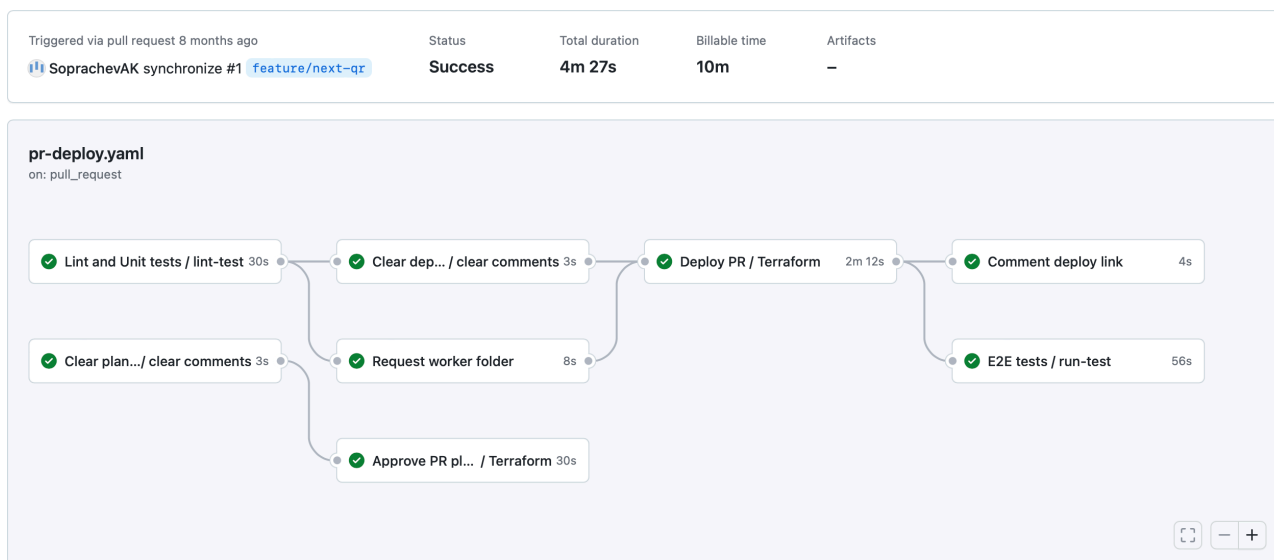


Рис. 7. Пайплайн в GitHub Actions




В Yandex Cloud изолированные окружения внутри одной организации называются директориями, из-за ограничений облачного провайдера, процесс создание новой директории занимает около 5 минут. Для ускорения развёртывания тестового окружения, был использован подход с использованием пула директорий. Во время развёртывания тестового окружения, GitHub Actions запрашивает свободную директорию, если такой нет, то создаётся новая, иначе используется уже существующая. После принятия Pull Request'a, директория очищается и возвращается в пул свободных.

Статический анализатор и Unit тесты запускаются на первом шаге на основе исходного кода без необходимости использования инфраструктуры. Это позволяет быстро выявлять ошибки в коде, и не тратить время на развёртывание тестового окружения. End-to-end[21] тесты (REST запросы в случае бекенда, и UI тесты в случае фронтенда) запускаются на тестовом окружении, что позволяет проверить работу приложения в реальном окружении.

Lint and Unit tests / lint-test summary

...





### ESLint

 Error	 Warning	 Total
0	2	2







▼ Warnings

- <https://github.com/umap-space/qr-generator/blob/99fff48aff0d708191899ab3f355fc023a64ab4c/app/src/utlis/gennext/generator.ts>
  - 108:17 'isDark' is assigned a value but never used. @typescript-eslint/no-unused-vars
- <https://github.com/umap-space/qr-generator/blob/99fff48aff0d708191899ab3f355fc023a64ab4c/app/tests/qrgenerator/qrgen.test.ts>
  - 1:10 'generate' is defined but never used. @typescript-eslint/no-unused-vars

### JEST

Result	Passed 	Failed 	Duration 
Passing 	6	0	3.408 s

▼ Tests







test	passed
QR Code > classifyRect	
QR Code > determinateLineSegments	
QR Code > generatePath	
QR Code > image must be square	
QR Code > svg image must have height and weidth	
sum moduleA > empty	

Job summary generated at run-time

E2E tests / run-test summary

...

### Cypress Results

Result	Passed 	Failed 	Pending 	Skipped 	Duration 
Passing 	28	0	0	0	13.432s

Job summary generated at run-time

Рис. 8. Отчёт о прохождении GitHub Actions

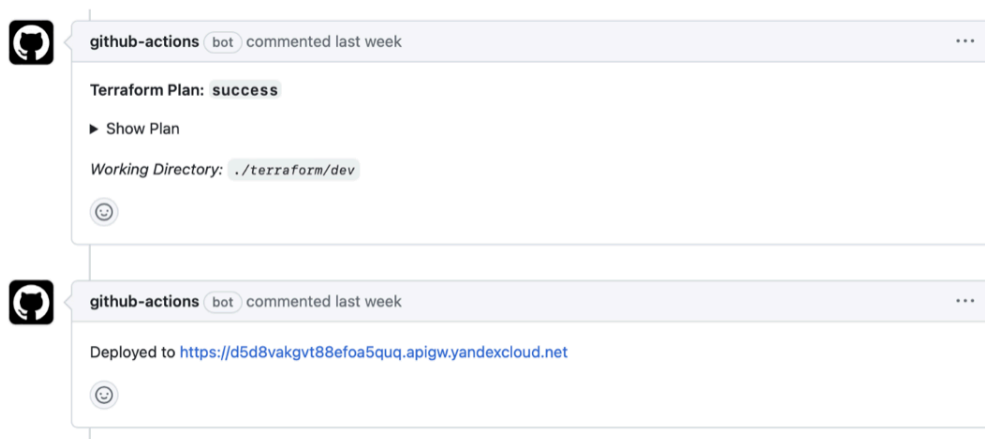


Рис. 9. Комментарий в Pull Request'e

После завершения пайплайна, в Pull Request'e создаётся комментарий с ссылкой на тестовое окружение, и выводом команды `terraform plan`. Это позволяет быстро проверить работу приложения, и убедиться в корректности изменений.



## СПИСОК ЛИТЕРАТУРЫ

1. Приложение PolyMap [Электронный ресурс]. URL: <https://apps.apple.com/ru/app/polymap/id1589702536> (дата обращения: 28.02.2025).
2. Что такое SaaS (ПО как услуга) [Электронный ресурс]. URL: <https://www.oracle.com/cis/applications/what-is-saas/> (дата обращения: 28.02.2025).
3. Yandex Appmetrica [Электронный ресурс]. URL: <https://appmetrica.yandex.ru/> (дата обращения: 28.02.2025).
4. Docker Develop faster. Run anywhere [Электронный ресурс]. URL: <https://www.docker.com/> (дата обращения: 28.02.2025).
5. Что такое Kubernetes [Электронный ресурс]. URL: <https://kubernetes.io/ru/> (дата обращения: 28.02.2025).
6. Бессерверные вычисления на AWS [Электронный ресурс]. URL: <https://aws.amazon.com/ru/serverless/> (дата обращения: 28.02.2025).
7. Облачная платформа Yandex Cloud [Электронный ресурс]. URL: <https://cloud.yandex.ru/> (дата обращения: 28.02.2025).
8. Yandex Serverless Containers [Электронный ресурс]. URL: <https://cloud.yandex.ru/services/serverless-containers> (дата обращения: 28.02.2025).
9. Yandex API Gateway [Электронный ресурс]. URL: <https://cloud.yandex.ru/services/api-gateway> (дата обращения: 28.02.2025).
10. Yandex Object Storage [Электронный ресурс]. URL: <https://cloud.yandex.ru/services/storage> (дата обращения: 28.02.2025).
11. What is Amazon S3 [Электронный ресурс]. URL: <https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html> (дата обращения: 28.02.2025).
12. Yandex Managed Service for YDB [Электронный ресурс]. URL: <https://cloud.yandex.ru/services/ydb> (дата обращения: 28.02.2025).
13. Для чего используют сервис CDN [Электронный ресурс]. URL: <https://selectel.ru/services/additional/cdn/> (дата обращения: 28.02.2025).
14. A global network built for the cloud [Электронный ресурс]. URL: <https://www.cloudflare.com/> (дата обращения: 28.02.2025).

15. WebSockets [Электронный ресурс]. URL: [https://developer.mozilla.org/ru/docs/Web/API/WebSockets\\_API](https://developer.mozilla.org/ru/docs/Web/API/WebSockets_API) (дата обращения: 28.02.2025).
16. Automate infrastructure on any cloud with Terraform [Электронный ресурс]. URL: <https://www.terraform.io/> (дата обращения: 28.02.2025).
17. Infrastructure as code [Электронный ресурс]. URL: <https://www.atlassian.com/microservices/cloud-computing/infrastructure-as-code> (дата обращения: 28.02.2025).
18. Git [Электронный ресурс]. URL: <https://git-scm.com/> (дата обращения: 28.02.2025).
19. Automate your workflow from idea to production [Электронный ресурс]. URL: <https://github.com/features/actions> (дата обращения: 28.02.2025).
20. Gitflow Workflow [Электронный ресурс]. URL: <https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow> (дата обращения: 28.02.2025).
21. Test. Automate. Accelerate With Cypress [Электронный ресурс]. URL: <https://www.cypress.io/> (дата обращения: 28.02.2025).