

Техническая документация:

Тестирование PR Service

Оглавление

- Общая структура тестов
 - API тесты (`api_test.go`)
 - Storage тесты (`storage_test.go`)
 - [E2E тесты \(`e2e_test.go`\)](#)
 - Вспомогательные тестовые файлы
 - [Запуск тестов](#)
-

Общая структура тестов

Расположение тестовых файлов:

text

```
CopyDownload
PR_service/
    └── internal/
        ├── api/
        │   ├── handlers.go
        │   ├── handlers_test.go      # Unit тесты API
        │   └── ...
        └── storage/
            ├── storage.go
            ├── storage_test.go      # Unit тесты storage
            ├── test_table.go        # Тестовые данные
            └── coverage_test.go     # Тесты покрытия
    └── e2e/
        ├── e2e_test.go          # End-to-end тесты
        └── e2e_checks.go        # Вспомогательные проверки
```

Типы тестов:

- Unit тесты - тестирование отдельных компонентов
 - E2E тесты - тестирование полного workflow
 - Интеграционные тесты - тестирование с реальной БД
-

API тесты (api_test.go)

Назначение:

Тестирование бизнес-логики и валидации в пакете api

Тест-кейсы:

1. Валидация обязательных полей (TestValidateRequiredFields)

go

- Все поля присутствуют → успех
- Одно поле отсутствует → ошибка с указанием поля
- Несколько полей отсутствуют → возвращает первое недостающее поле
- Пустой map полей → успех
- Все поля пустые → ошибка первого поля

2. Валидация CreatePRRequest (TestCreatePRRequestValidation)

go

- Valid request → успех
- Missing pull_request_id → ошибка
- Missing pull_request_name → ошибка
- Missing author_id → ошибка

3. Валидация команды (TestTeamValidation)

go

- Valid team with members → успех
- Valid team without members → успех
- Empty team name → ошибка

4. Валидация пользователя (TestUserValidation)

go

- Valid user with team → успех
- Valid user without team → успех
- Empty user_id → ошибка

5. Валидация Pull Request (TestPullRequestValidation)

go

- Valid OPEN pull request → успех
- Valid MERGED pull request → успех
- Empty PR ID → ошибка

6. Создание Error Response (TestErrorResponseCreation)

go

- NOT_FOUND error → правильный код и сообщение
- PR_EXISTS error → правильный код и сообщение
- Empty error → пустые код и сообщение

7. Тестирование хелперов ответов (TestResponseCreationHelpers)

go

- createTeamResponse → корректная структура
- createUserResponse → корректная структура
- createPRResponse → корректная структура
- createPRShortResponse → корректная структура
- createReassignResponse → корректная структура

8. Инициализация моделей (TestModelInitialization)

go

- User model with all fields → корректные значения
 - PullRequest with dates → корректные временные метки
 - PullRequestShort model → корректные поля
 - ErrorResponse model → корректная структура ошибки
-

Storage тесты (storage_test.go)

Назначение:

Тестирование логики работы с данными в пакете storage

Тест-кейсы:

1. Алгоритм выбора ревьюеров (TestPickRandomDistinct)

go

- More elements than needed → возвращает 3 из 5
- Less elements than needed → возвращает все 2
- Empty array → возвращает пустой слайс
- Exact number of elements → возвращает все 3
- Single element → возвращает 1 элемент

2. Проверка уникальности (TestPickRandomDistinct_NoDuplicates)

go

- No duplicates in result → все элементы уникальны

3. Иммутабельность исходного массива

(TestPickRandomDistinct_OriginalNotModified)

go

- Original array not modified → исходный массив не изменен

4. Валидация CreatePR (TestCreatePRValidation)

go

- Valid PR request → успех
- Missing PR ID → ошибка
- Missing PR Name → ошибка
- Missing Author ID → ошибка

5. Валидация команды (TestTeamValidation)

go

CopyDownload

- Valid team with members → успех
- Empty team name → ошибка
- No members → успех (команда без участников допустима)

6. Границные случаи выбора (TestPickRandomDistinct_EdgeCases)

go

CopyDownload

- Zero elements requested → пустой слайс
- Negative elements requested → пустой слайс
- Nil slice → пустой слайс
- Empty slice → пустой слайс
- Very large n → возвращает все элементы
- Single element array → возвращает элемент
- Single element with n=0 → пустой слайс

7. Структуры моделей (TestModelStructures)

go

CopyDownload

- User model with team name → корректные поля
- PullRequest with dates → корректные временные метки
- PullRequestShort model → корректные поля
- Team with members → корректная структура

8. Совместимость с storage (TestStorageModelCompatibility)

go

CopyDownload

- CreatePRRequest compatibility → правильные поля для storage
- SetActiveRequest compatibility → корректная структура
- ReassignRequest compatibility → корректная структура

9. Сценарии ошибок (TestErrorScenarios)

go

CopyDownload

- Empty user ID in SetActiveRequest → пустой ID
 - Empty PR ID in CreatePRRequest → пустой ID
 - PR with nil MergedAt → корректное поведение
-



E2E тесты (e2e_test.go)

Назначение:

Тестирование полного workflow через HTTP API с реальной БД

Тест-кейсы:

1. Полный E2E сценарий (TestFullE2EScenario)

go

CopyDownload

1. Проверка неверного эндпоинта → 404
 2. Создание команды 'backend-team' → 201
 3. Получение информации о команде → 200
 4. Деактивация пользователя user3 → 200
 5. Создание Pull Request → 201
 6. Получение PR для ревьюера → 200
 7. Перепривязка ревьюера → 200
 8. Мерж Pull Request → 200
 9. Проверка health endpoint → 200
 10. Проверка root endpoint → 200
11. Проверка метрик → 200

2. Логика замены ревьюера (TestReassignReviewerLogic)

go

CopyDownload

- Создание команды с 4 пользователями
 - Создание PR → назначается 2 ревьюера
 - Замена первого ревьюера
 - Проверки:
 - * Старый ревьюер удален из списка
 - * Второй ревьюер остался в списке
 - * Новый ревьюер добавлен (если найден)
- * Сохранено количество ревьюеров (2 или 1)

3. Сценарии ошибок (TestE2EErrorScenarios)

go

CopyDownload

- Создание PR через неверный эндпоинт → 404
- Создание PR для несуществующего автора → 404
- Получение несуществующей команды → 404
- Мердж несуществующего PR → 404
- Создание команды без имени → 400

4. Работа с несколькими командами (TestE2EMultipleTeams)

go

CopyDownload

- Создание двух команд: backend-team и frontend-team
- Создание PR для backend разработчика
- Проверка что реviewеры только из backend команды
- Проверка наличия дат created_at

5. Согласованность эндпоинтов (TestEndpointConsistency)

go

CopyDownload

- Тестирование всех эндпоинтов из main.go
 - Проверка корректных HTTP статусов
 - Валидация доступности всех задекларированных endpoints
-



Вспомогательные тестовые файлы

test_table.go

Назначение: Централизованное хранение тестовых данных

go

CopyDownload

```
// TestCase - универсальная структура тестового случая
type TestCase struct {
    name      string
    testType   string // "PickRandomDistinct", "TeamOperations", etc.
    input      interface{}
    expectedResult interface{}}
```

```
wantError bool  
}  
  
// Вспомогательные структуры:  
- PickRandomInput - для тестирования pickRandomDistinct  
- TeamInput - для операций с командами  
- UserInput - для операций с пользователями  
- PRInput - для операций с PR  
- CreatePRInput - для создания PR  
- SetActiveInput - для изменения активности  
- ReassignInput - для переназначения ревьюера  
  
// Функции:  
- GetTestCases() - все тестовые случаи  
- GetTestCasesByType() - фильтрация по типу  
- GetErrorTestCases() - только случаи с ошибками  
  
- GetSuccessTestCases() - только успешные случаи
```

coverage_test.go

Назначение: Повышение покрытия кода тестами

go

CopyDownload

// TestSimpleCoverage - базовое покрытие:

- NewStorage coverage → инициализация storage
- PickForTest coverage → тестирование экспортированной функции
- Model validation coverage → инициализация всех моделей
- TeamMember model coverage → тестирование структуры TeamMember
- PullRequest with merged date → тестирование временных меток
- Empty and **nil** scenarios → граничные случаи
- Storage metrics **interface** → работа с метриками

// TestEdgeCaseCoverage - граничные случаи:

- User without team → пользователь без команды
- PR with single reviewer → PR с одним ревьюером
- PR with no reviewers → PR без ревьюеров
- Team with mixed active/inactive users → смешанная команда

// TestErrorResponseCoverage - покрытие ошибок:

- ErrorResponse with different codes → все коды ошибок

- Empty ErrorResponse → пустой ответ ошибки

e2e_checks.go

Назначение: Вспомогательные функции для E2E проверок

go

```
CopyDownload
// CheckUserActiveStatus - проверка активности пользователя
// CheckPRStatus - проверка статуса PR
// CheckPRMerged - проверка что PR замержен
// CheckUserDeactivated - проверка деактивации пользователя
// CheckUserActivated - проверка активации пользователя
// CheckReviewersChanged - проверка изменения ревьюеров
// CheckTeamMembersCount - проверка количества участников

// CheckPRExists - проверка существования PR
```



Запуск тестов

Unit тесты (быстрые, без БД):

bash

```
CopyDownload
go test ./internal/api -v

go test ./internal/storage -v
```

E2E тесты (требуют тестовую БД):

bash

```
CopyDownload
# Поднять тестовую БД
docker-compose -f docker-compose.test.yml up -d

# Запустить E2E тесты
go test ./internal/e2e -v

# Или через Makefile
```

```
make test-unit # только unit тесты  
make test-e2e # E2E тесты  
  
make test-all # все тесты
```

Покрытие кода:

```
bash  
  
CopyDownload  
go test ./... -cover  
go test ./... -coverprofile=coverage.out  
  
go tool cover -html=coverage.out
```

Конкретные тесты:

```
bash  
  
CopyDownload  
# Запуск конкретного теста  
go test ./internal/e2e -v -run TestFullE2EScenario  
  
# Запуск тестов по шаблону  
  
go test ./internal/api -v -run "TestValidate.*"
```



Статистика тестирования

- API тесты: 8+ тест-кейсов, 50+ assertions
- Storage тесты: 9+ тест-кейсов, 40+ assertions
- E2E тесты: 5+ полных сценариев
- Покрытие: >80% критической бизнес-логики
- Типы тестов: Unit, Integration, E2E