Техническая документация: PR Service

Оглавление

Архитектура проекта

Структура файлов:

```
PR_service/
├── cmd/server/main.go      # Альтернативная точка входа
├── internal/
│   ├── api/
│   │   ├── handlers.go     # HTTP обработчики
│   │   ├── middleware.go   # Таймауты и метрики
│   │   ├── metrics.go      # Prometheus метрики
│   │   └── utils.go        # Вспомогательные функции
│   ├── storage/
│   │   └── storage.go      # Работа с PostgreSQL
│   └── models/
│       └── models.go       # Модели данных
├── docker-compose.yml      # Продакшен окружение

└── docker-compose.test.yml # Тестовое окружение
```

Модели данных:

```
type Team struct {
   TeamName string `json:"team_name"`
   Members  []User `json:"members"`
}

type User struct {
```

```
    UserID   string `json:"user_id"`
    Username string `json:"username"`
    TeamName string `json:"team_name"`
    IsActive bool   `json:"is_active"`
}

type PullRequest struct {
    PullRequestID   string    `json:"pull_request_id"`
    PullRequestName string    `json:"pull_request_name"`
    AuthorID        string    `json:"author_id"`
    Status          string    `json:"status"` // OPEN|MERGED
    Reviewers       []string  `json:"assigned_reviewers"`
    CreatedAt       time.Time `json:"createdAt,omitempty"`
    MergedAt        *string   `json:"mergedAt,omitempty"`

}
```

API Endpoints

Системные endpoints

Метод	Endpoint	Описание
`GET`	`/`	Информация о сервисе
`GET`	`/health`	Health check
`GET`	`/metrics`	Prometheus метрики
`GET`	`/metrics/data`	Детальные метрики

Управление командами

Метод	Endpoint	Описание
`POST`	`/team/add`	Создать/обновить команду

`GET`	`/team/get?team_name=NAME`	Получить информацию о команде

Управление пользователями

Метод	Endpoint	Описание
`POST`	`/users/setIsActive`	Активировать/деактивировать пользователя
`GET`	`/users/getReview?user_id=ID`	Получить PR для ревью пользователя

Управление Pull Requests

Метод	Endpoint	Описание
`POST`	`/pullRequest/create`	Создать PR
`POST`	`/pullRequest/merge`	Замержить PR
`POST`	`/pullRequest/reassign`	Заменить ревьюера

Бизнес-логика

Создание команды (`/team/add`)

- Создает команду если не существует
- Добавляет пользователей в команду
- Обновляет существующих пользователей
- Автоматически связывает пользователей с командой через `team_name`

Создание PR (`/pullRequest/create`)

1. Валидация автора - автор должен существовать и быть активным
2. Проверка команды - автор должен состоять в команде

3. Выбор ревьюеров - случайно выбирает 2 активных пользователя из той же команды
4. Исключения - не выбирает автора и неактивных пользователей
5. Создание PR - устанавливает статус "OPEN", время создания

Замена ревьюера (`/pullRequest/reassign`)

1. Проверка PR - PR должен существовать и быть в статусе "OPEN"
2. Проверка ревьюера - старый ревьюер должен быть назначен на PR
3. Поиск замены - ищет активного пользователя из той же команды
4. Замена - удаляет старого, добавляет нового ревьюера
5. Возврат - возвращает обновленный PR и ID нового ревьюера

Мерж PR (`/pullRequest/merge`)

1. Проверка статуса - если уже мерджен, возвращает текущее состояние
2. Обновление - меняет статус на "MERGED", устанавливает время мержа
3. Блокировка - после мержа нельзя менять ревьюеров

Запуск и конфигурация

Рекомендуемый быстрый запуск:

```
# 1. Клонировать репозиторий
git clone https://github.com/NovvAleA/PullRequesService

# 2. Перейти в папку
cd PullRequesService/PR_service

# 3. Запустить сервис
docker-compose up
```

Локальный запуск:

bash

```
# Запуск с Docker Compose
docker-compose up -d
```

```
# Или сборка и запуск вручную
docker build -t pr-service .

docker run -p 8080:8080 pr-service
```

Переменные окружения:

```
bash
DATABASE_URL=postgres://pguser:password@localhost:5432/pr_reviewer_db?sslmode=disable

PORT=8080
```

Примеры использования

1. Создание команды разработчиков

```
bash
curl -X POST http://localhost:8080/team/add \
 -H "Content-Type: application/json" \
 -d '{
   "team_name": "backend-team",
   "members": [
     {
       "user_id": "user1",
       "username": "Алексей Петров",
       "is_active": true
     },
     {
       "user_id": "user2",
       "username": "Мария Сидорова",
       "is_active": true
     },
     {
       "user_id": "user3",
       "username": "Иван Иванов",
       "is_active": true
     },
```

```
    {
      "user_id": "user4",
      "username": "Елена Смирнова",
      "is_active": true
    }
  ]

}'
```

2. Получение информации о команде

bash

```
curl "http://localhost:8080/team/get?team_name=backend-team"
```

3. Деактивация пользователя

bash

```
curl -X POST http://localhost:8080/users/setIsActive \
  -H "Content-Type: application/json" \
  -d '{
    "user_id": "user3",
    "is_active": false

}'
```

4. Создание Pull Request

bash

```
curl -X POST http://localhost:8080/pullRequest/create \
  -H "Content-Type: application/json" \
  -d '{
    "pull_request_id": "pr-backend-001",
    "pull_request_name": "Реализация аутентификации",
    "author_id": "user1"

}'
```

5. Получение PR для ревьюера

bash

```
curl "http://localhost:8080/users/getReview?user_id=user2"
```

6. Замена ревьюера

bash

```
curl -X POST http://localhost:8080/pullRequest/reassign \
 -H "Content-Type: application/json" \
 -d '{
   "pull_request_id": "pr-backend-001",
   "old_user_id": "user2"

 }'
```

7. Мерж Pull Request

bash

```
curl -X POST http://localhost:8080/pullRequest/merge \
 -H "Content-Type: application/json" \
 -d '{
   "pull_request_id": "pr-backend-001"

 }'
```

8. Проверка здоровья сервиса

bash

```
curl http://localhost:8080/health
```

9. Просмотр метрик

bash

```
curl http://localhost:8080/metrics/data
```

---

10. Health Check endpoints:

- `GET /health` - проверка БД и системных ресурсов
- `GET /metrics` - Prometheus метрики
- `GET /metrics/data` - детальные бизнес-метрики

Метрики и мониторинг

Сервис предоставляет метрики:

- HTTP запросы - количество, длительность, статусы
- Бизнес-метрики - созданные PR, мержи, ошибки
- Database метрики - время выполнения запросов
- Системные метрики - goroutines, память, uptime

Все метрики доступны в формате Prometheus на `/metrics` и в JSON на `/metrics/data`.

Ограничения и особенности

- Максимум 2 ревьюера на PR
- Автор не может быть ревьюером своего PR
- Неактивные пользователи не назначаются ревьюерами
- После мержа PR нельзя менять ревьюеров
- Все операции атомарны (используются транзакции)
- Таймаут запросов: 300ms