Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ

к лабораторной работе №2

По курсу: «Тестирование и отладка ПО»

*Тема: **«Интеграционное и E2E тестирование»***

Студент ИУ7-75Б
Афимин И. Е.

Преподаватель
Жаров С. В.

Москва, 2021 г.

Результаты выполнения ЛР

1) Запуск CI/CD в GitHub Actions

```
name: Simple Build
on:
  push:
    branches: [master]
  pull_request:
    branches: [master]

jobs:
  tests:
    runs-on: ubuntu-latest
    services:                        -- Поднимаю сервер Постгреса
      postgres:
        image: postgres:12
        env:
          POSTGRES_USER: mdb
          POSTGRES_PASSWORD: mdb
          POSTGRES_DB: mdb
        ports:
          - 5432:5432
        options: >-
          --health-cmd pg_isready
          --health-interval 10s
          --health-timeout 5s
          --health-retries 5

    steps:
      - name: Install Go
        uses: actions/setup-go@v2
        with:
          go-version: 1.16.x

      - name: Checkout code
        uses: actions/checkout@v2

      - name: Create DataBase         -- Запускаю скрипт создания таблиц в БД
        env:
          DB_HOST: localhost
          DB_USER: mdb
          DB_PASSWORD: mdb
          DB_NAME: mdb
          DB_PORT: 5432
          DB_DIALECT: postgres
          PGPASSWORD: mdb
        run: psql -d postgresql://mdb@localhost/mdb -f install.sql

      - name: UnitTest                -- Запускаю Unit тесты
        run: |
          go test -coverprofile=coverage1.out -coverpkg=./... -cover ./... &&
cat coverage1.out | grep -v _mock | grep -v pb.go > cover.out &&go tool cover
-func=cover.out

      - name: IntegrationTests.     -- Запускаю Интеграционные тесты
        run: |
          cd internal/ratings/integration
        go test

    - name: E2ETests            -- Запускаю E2E тесты
      run: |
        cd internal/ratings/e2e
        go test
```

2) Интеграционные тесты. Тестируются CRUD операции при подключении к реальной БД без заглушек.

```go
package e2e_test

import (
	"context"
	"fmt"
	ratingsDBStorage "github.com/go-park-mail-ru/2021_1_kekEnd/internal/ratings/repository/dbstorage"
	ratingsUseCase "github.com/go-park-mail-ru/2021_1_kekEnd/internal/ratings/usecase"
	"github.com/jackc/pgx/v4/pgxpool"
	"github.com/stretchr/testify/assert"
	"log"
	"os"
	"testing"
)


func TestRating(t *testing.T) {
	os.Setenv("DB_CONNECT", "postgres://mdb:mdb@127.0.0.1:5432/mdb")

	connStr, connected := os.LookupEnv("DB_CONNECT")
	if !connected {
		fmt.Println(os.Getwd())
		log.Fatal("Failed to read DB connection data")
	}
	dbpool, err := pgxpool.Connect(context.Background(), connStr)
	assert.NoError(t, err)

	t.Run("CreateRating and GetRating", func(t *testing.T) {
		_, err = dbpool.Exec(context.Background(), "TRUNCATE TABLE mdb.movie_rating")
		assert.NoError(t, err)

		ratingsRepo := ratingsDBStorage.NewRatingsRepository(dbpool)
		ratingsUC := ratingsUseCase.NewRatingsUseCase(ratingsRepo)

		err = ratingsUC.CreateRating("user1", "2", 4)
		assert.NoError(t, err)

		rating, err := ratingsUC.GetRating("user1", "2")
		assert.NoError(t, err)

		assert.Equal(t, rating.UserID, "user1")
		assert.Equal(t, rating.MovieID, "2")
		assert.Equal(t, rating.Score, 4)
	})

	t.Run("CreateRating and GetRating", func(t *testing.T) {
		_, err = dbpool.Exec(context.Background(), "TRUNCATE TABLE mdb.movie_rating")
		assert.NoError(t, err)

		ratingsRepo := ratingsDBStorage.NewRatingsRepository(dbpool)
		ratingsUC := ratingsUseCase.NewRatingsUseCase(ratingsRepo)

		err = ratingsUC.CreateRating("user1", "2", 4)
		assert.NoError(t, err)

		ratingsRepo2 := ratingsDBStorage.NewRatingsRepository(dbpool)
		ratingsUC2 := ratingsUseCase.NewRatingsUseCase(ratingsRepo2)

		err = ratingsUC2.CreateRating("user1", "2", 4)
		assert.Error(t, err)

	})

	t.Run("UpdateRating", func(t *testing.T) {
		_, err = dbpool.Exec(context.Background(), "TRUNCATE TABLE mdb.movie_rating")
		assert.NoError(t, err)

		ratingsRepo := ratingsDBStorage.NewRatingsRepository(dbpool)
		ratingsUC := ratingsUseCase.NewRatingsUseCase(ratingsRepo)

		err = ratingsUC.CreateRating("user1", "2", 4)
		assert.NoError(t, err)

		rating, err := ratingsUC.GetRating("user1", "2")
		assert.NoError(t, err)

		assert.Equal(t, rating.UserID, "user1")
		assert.Equal(t, rating.MovieID, "2")
		assert.Equal(t, rating.Score, 4)

		err = ratingsUC.UpdateRating("user1", "2", 8)
		assert.NoError(t, err)

		ratingAfterUpdate, err := ratingsUC.GetRating("user1", "2")
		assert.NoError(t, err)

		assert.Equal(t, ratingAfterUpdate.UserID, "user1")
		assert.Equal(t, ratingAfterUpdate.MovieID, "2")
		assert.Equal(t, ratingAfterUpdate.Score, 8)
	})

	t.Run("DeleteRating", func(t *testing.T) {
		_, err = dbpool.Exec(context.Background(), "TRUNCATE TABLE mdb.movie_rating")
		assert.NoError(t, err)

		ratingsRepo := ratingsDBStorage.NewRatingsRepository(dbpool)
		ratingsUC := ratingsUseCase.NewRatingsUseCase(ratingsRepo)

		err = ratingsUC.CreateRating("user1", "2", 4)
		assert.NoError(t, err)

		rating, err := ratingsUC.GetRating("user1", "2")
		assert.NoError(t, err)

		assert.Equal(t, rating.UserID, "user1")
		assert.Equal(t, rating.MovieID, "2")
		assert.Equal(t, rating.Score, 4)

		err = ratingsUC.DeleteRating("user1", "2")
		assert.NoError(t, err)

		_, err = ratingsUC.GetRating("user1", "2")
		assert.Error(t, err)

	})
}
```

3) E2E тестирование

Тестируются CRUD операции для работы с рейтингом. Используются реальные запросы к серверу.
Запускается сервер, и дальше на него отправляются HTTP-запросы.

```go
package e2e_test

import (
	"context"
	"fmt"
	"github.com/go-park-mail-ru/2021_1_kekEnd/internal/server"
	constants "github.com/go-park-mail-ru/2021_1_kekEnd/pkg/const"
	"github.com/jackc/pgx/v4/pgxpool"
	"github.com/labstack/echo"
	"github.com/stretchr/testify/assert"
	"io/ioutil"
	"log"
	"net/http"
	"os"
	"strings"
	"testing"
)

func TestE2ERating(t *testing.T) {
	os.Setenv("DB_CONNECT", "postgres://mdb:mdb@127.0.0.1:5432/mdb")

	connStr, connected := os.LookupEnv("DB_CONNECT")
	if !connected {
		fmt.Println(os.Getwd())
		log.Fatal("Failed to read DB connection data")
	}
	dbpool, err := pgxpool.Connect(context.Background(), connStr)
	assert.NoError(t, err)


	app := server.NewApp()

	go func() {
		err := app.Run(constants.Port)
		assert.NoError(t, err)
	}()

	t.Run("CreateRating", func(t *testing.T) {
		_, err = dbpool.Exec(context.Background(), "TRUNCATE TABLE mdb.movie_rating")
		assert.NoError(t, err)


		// AddRating
		reqStrAddRating := `{"movie_id":"1", "score": "4"}`
		reqAddRating, err := http.NewRequest(echo.POST, fmt.Sprintf("http://localhost:%s/api/v1/ratings", constants.Port), strings.NewReader(reqStrAddRating))
		assert.NoError(t, err)

		reqAddRating.Header.Set(echo.HeaderContentType, echo.MIMEApplicationJSON)

		client := http.Client{}
		response, err := client.Do(reqAddRating)
		assert.NoError(t, err)
		assert.Equal(t, http.StatusCreated, response.StatusCode)


		// GetRating
		reqGetRating, err := http.NewRequest(echo.GET, fmt.Sprintf("http://localhost:%s/api/v1/ratings/%d", constants.Port, 1), strings.NewReader(""))
		assert.NoError(t, err)

		reqGetRating.Header.Set(echo.HeaderContentType, echo.MIMEApplicationJSON)

		clientGetRating := http.Client{}
		responseGetRating, err := clientGetRating.Do(reqGetRating)
		assert.NoError(t, err)
		assert.Equal(t, http.StatusOK, responseGetRating.StatusCode)

		byteBodyGetRating, err := ioutil.ReadAll(responseGetRating.Body)
		assert.NoError(t, err)

		assert.Equal(t, `{"username":"user1","movie_id":"1","score":4}`, strings.Trim(string(byteBodyGetRating), "\n"))
		response.Body.Close()


		// UpdateRating
		reqStrUpdateRating := `{"movie_id":"1", "score": "8"}`
		reqUpdateRating, err := http.NewRequest(http.MethodPut, fmt.Sprintf("http://localhost:%s/api/v1/ratings", constants.Port), strings.NewReader(reqStrUpdateRating))
		assert.NoError(t, err)


```

```
                reqUpdateRating.Header.Set(echo.HeaderContentType,
echo.MIMEApplicationJSON)

                clientUpdateRating := http.Client{}
                responseUpdateRating, err :=
clientUpdateRating.Do(reqUpdateRating)
                assert.NoError(t, err)
                assert.Equal(t, http.StatusOK,
responseUpdateRating.StatusCode)


                // GetRatingAfterUpdate
                clientGetRatingAfterUpdate := http.Client{}
                responseGetRatingAfterUpdate, err :=
clientGetRatingAfterUpdate.Do(reqGetRating)
                assert.NoError(t, err)
                assert.Equal(t, http.StatusOK,
responseGetRatingAfterUpdate.StatusCode)

                byteBodyGetRatingAfterUpdate, err :=
ioutil.ReadAll(responseGetRatingAfterUpdate.Body)
                assert.NoError(t, err)

                assert.Equal(t,
`{"username":"user1","movie_id":"1","score":8}`,
strings.Trim(string(byteBodyGetRatingAfterUpdate), "\n"))
                response.Body.Close()


                // DeleteRating
                reqDeleteRating, err := http.NewRequest(http.MethodDelete,
fmt.Sprintf("http://localhost:%s/api/v1/ratings/%d", constants.Port, 1),
strings.NewReader(""))
                assert.NoError(t, err)

                reqDeleteRating.Header.Set(echo.HeaderContentType,
echo.MIMEApplicationJSON)

                clientDeleteRating := http.Client{}
                responseDeleteRating, err :=
clientDeleteRating.Do(reqDeleteRating)
                assert.NoError(t, err)
                assert.Equal(t, http.StatusOK,
responseDeleteRating.StatusCode)


                // GetRatingAfterDelete
                clientGetRatingAfterDelete := http.Client{}
                responseGetRatingAfterDelete, err :=
clientGetRatingAfterDelete.Do(reqGetRating)
                assert.NoError(t, err)
                assert.Equal(t, http.StatusNotFound,
responseGetRatingAfterDelete.StatusCode)

                byteBodyGetRatingAfterDelete, err :=
ioutil.ReadAll(responseGetRatingAfterDelete.Body)
                assert.NoError(t, err)
                assert.Equal(t, "",
strings.Trim(string(byteBodyGetRatingAfterDelete), "\n"))
                response.Body.Close()
        })
}
```