федеральное государственное автономное образовательное учреждение высшего образования

Санкт-Петербургский национальный исследовательский УНИВЕРСИТЕТ информационных технологий, механики и оптики

**Факультет программной инженерии и компьютерной техники**

**ЛАБОРАТОРНАЯ РАБОТА №2**

**ПО ДИСЦИПЛИНЕ «ТЕСТИРОВАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ»**

Выполнил: Чудаков М.И.

Группа: P3319

Преподаватель: Исаев И.В.

Санкт-Петербург

2019

**Цель**

Необходимо разработать консольное приложение в соответствие с методологий TDD.

**Задачи**

1. Необходимо из краткого описания сформировать функциональные требования к программному продукту (не менее 20, т.е. дополнить);
2. Необходимо полностью покрыть тестами данные требования используя разработку через тестирование;
3. Для хранения данных использовать PostgreSQL или MongoDB;
4. Реализовать продукт 100% покрытый тестами.

**Вариант 19 (4)**

Разработка консольного приложения “Музыкальный каталог”.

При запуске программы пользователю предлагается справочная информация по использованию. Далее предлагается ввести команду, позволяющую выполнить одно из действий:

* осуществить поиск музыкальной композиции в каталоге по определенному критерию;
* вывести информацию о всех существующих в каталоге композициях;
* добавить информацию о композиции в каталог;
* удалить существующую в каталоге запись;
* выйти из программы.

Критериями поиска могут служить: имя (название) автора/исполнителя или название композиции. В качестве результата поиска в консоль должен выводиться список композиций в виде «исполнитель – название». Удаление или добавление записи осуществляется после ввода всей информации о композиции.

**Сформулированные функциональные требования**

1. При запуске приложение предоставляет справочную информацию по использованию.
2. Приложение предоставляет возможность выйти из программы, для этого пользователю необходимо ввести команду “выйти”.
3. Музыкальная композиция содержит следующие критерии: название, автор, жанр. Каждый критерий является отдельной сущностью БД: Композиция (название, автор, жанр); Автор (имя); Жанр (название);
4. Все данные хранятся в БД под управлением СУБД postgresql
5. Сочетание названия и автора композиции должно быть уникальным. Имя автора должно быть уникальным.
6. Добавление новой композиции в каталог.
7. Добавление нового жанра.
8. При добавлении новой композиции пользователь последовательно вводит название, автора и жанр. В случае если введенный автор существует, данные в таблице “автор”, не дублируются, при создании композиции, в поле “автор” заносится существующая запись. В случае если пользователь ввел не существующий жанр выводится сообщение об ошибке с предложение повторно ввести название жанра. В сообщении об ошибке пользователю выводятся примеры существующий названий жанров (не более 5), предыдущие введенные данные сохраняются.
9. Вывести информацию о всех существующих композициях.
10. Вывести все композиции одного автора, имя автора вводится пользователем.
11. Найти композицию по названию, в случае если названия совпадают выводит все.
12. Удалить существующую в каталоге запись.
13. Поиск композиций по жанру
14. Вывести информацию о всех существующих авторах.
15. Вывести информацию о всех существующих жанров.
16. Создание плейлиста
17. Добавление существующей композиции в плейлист
18. Удаление композиции из плейлиста
19. Возможность удалить плейлист
20. Просмотр всех композиций в плейлисте
21. При переходе в любой блок меню пользователю выводится информация в виде списка названий опций, каждая опция – переход в другой блок меню. Для перехода пользователю необходимо ввести название соответствующей опции.
22. Блок меню с добавлением новой композиции предлагает пользователю ввести новую композицию, сохранить внесенные изменения, отменить. После добавления новой композиции (смотреть пункт 8), пользователь переходит в этот же пункт меню. Для сохранения изменений необходимо ввести команду “сохранить внесенные изменения”, только после этого все данные сохраняются в БД.