МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по производственной технологической практике

Тема: Изучение современных методов и технологий разработки и программирования

Студентка гр. 8303 _____ Лисок М.А.

Руководитель _____ Шабунина Е.И.

Санкт-Петербург

2021

ЗАДАНИЕ

НА ПРОИЗВОДСТВЕННУЮ ТЕХНОЛОГИЧЕСКУЮ ПРАКТИКУ

Студентка: Лисок М.А.

Группа 8303

Тема практики: Изучение современных методов и технологий разработки и программирования

Задание на практику:

Ознакомиться с методологиями разработки ПО и системами контроля версий. Узнать подробнее о компетенциях и обязанностях бизнес-аналитиков, разработчиков ПО, специалистов по автоматическому тестированию. Разработать приложение из предложенного списка для платформы Android.

Сроки прохождения практики: 15.02.2021 – 17.07.2021

Дата сдачи отчета: 17.07.2021

Дата защиты отчета: 17.07.2021

Студентка _____ Лисок М.А.

Руководитель _____ Шабунина Е.И.

АННОТАЦИЯ

Целью работы является усвоение и закрепление на практике знаний методологии Agile, системы Git. Разобраться в функциях профессий тестировщика и бизнес-аналатика при разработке ПО. Изучить программные платформы android, .NET. Для закрепления пройденного материала написать приложение таймера для android.

SUMMARY

The aim of the work is to assimilate and consolidate in practice the knowledge of the Agile methodology, the Git system. Understand the functions of the professions of a tester and business analyst in software development. Explore the android, .NET software platforms. To consolidate the passed material, write a timer application for android.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель производственной практики – формирование первичных профессиональных знаний и навыков, полученных при теоретическом обучении. Также производственная практика способствует освоению общепрофессиональных и профессиональных дисциплин, формированию у обучающихся общих представлений о технологических процессах, оборудовании.

Задачи практики:

- Развитие практических навыков, формирование профессиональных компетенций и получение опыта профессиональной деятельности;
- Развитие навыков планирования, управления своим временем и прочими ресурсами;
- Развитие интереса и повышение мотивации к профессиональной деятельности, связанных с IT-технологиями;
- Формирование представлений о видах профессиональной деятельности и приобретение опыта в различных сферах профессиональной деятельности, связанных с IT-технологиями;
- Знакомство со структурой организации и проектами, которые реализуют компании в области информационных технологий;
- Знакомство с жизненным циклом продукта профессиональной деятельности, принципами менеджмента качества и маркетинга;
- Формирование профессионального мировоззрения и этики будущего специалиста по информационным технологиям и программированию.

1. ЦЕЛИ ПРОЕКТА

Цели проекта:

- Изучить подходы «гибкой разработки» на примере Agile;
- Ознакомиться с основными командами системы контроля версий git на примере GitHub;
- Изучить методы и подходы бизнес-аналитики к разработке программных продуктов;
- Получить опыт тестирования программного обеспечения;
- Познакомиться с разработкой приложения для Android;
- Получить опыт использования на практике изученных методов и подходов современной разработки и программирования.

2. ИСПОЛЬЗОВАННЫЕ ТЕХНОЛОГИИ И ОПИСАНИЕ ПРОЕКТА

2.1. Использованные технологии

Цель производственной практики – формирование первичных профессиональных знаний и навыков, полученных при теоретическом обучении. В качестве системы контроля версий git для изучения и применения в проекте использовался GitHub. При изучении тестирования API и графического веб интерфейса использовался стек технологий: Java11, Junit5, Spring, MavenApache, Selenium, Cucumber, Jenkins.

2.2. Основные требования к приложению:

- Приложение должно быть написано для платформы Android
- Таймер должен работать в фоновом режиме
- История таймера сохраняется
- По окончании таймера приходит уведомление

2.3. Реализация таймера:

Для реализации бизнес логики был написан фасад TimerFacade

```
public class TimerFacade implements Timer {
    private final Timer timer;
    private final Notifier notifier;
    private final Storage storage;
public TimerFacade(TimeContainer timeScheduled, Notifier
   notifier, Storage storage, TimerEventListenerBuilder
   eventListener) {

     eventListener.onFinish(this::finish);
     this.timer = new ThreadedTimerImpl(timeScheduled.

     toSeconds(), eventListener.build()); this.notifier =
notifier;
this.storage = storage;

}
```

По завершению работы таймера, запущенного в отдельном потоке происходит сохранение данных и отправка нотификации.

```
private void finish(TimerData data) {

     storage.save(data);

     notifier.sendNotification(data);

}
```

2.4. Реализация хранилища

Хранилища взаимозаменяемы, так как таймер работает с контрактом интерфейса Storage. Конкретная реализация хранилища, например, с помо- щью SharedPreferences выглядит примерно так:

```
public class SharedPreferencesStorage implements Storage {

      private final SharedPreferences preferences;
     private final static String PREFERENCES_ID = "TimerHistory";
     private final Logger LOGGER = Logger.getLogger("ru.etu.timer.

     ui.storage.SharedPreferencesStorage");

     public SharedPreferencesStorage(Context context) {
     preferences = context.getSharedPreferences(PREFERENCES_ID

     , Context.MODE_PRIVATE);

LOGGER.info("SharedPreferencesStorage has been created"); }

...
```

с методами вроде

```
@Override
public boolean save(TimerData timerData) {

     SharedPreferences.Editor editor = preferences.edit();
```

```
    try {
        String jsonRepresentation = timerData.toJsonString();
        LOGGER.info(String.format("Saving %s",
        jsonRepresentation)); e
        ditor.putString(timerData.getId(),
        jsonRepresentation);
      } catch (JSONException e) {
        e.printStackTrace();
       }
    return editor.commit();
}
```

2.5. Отделение бизнес логики от интерфейса

Для отделения бизнес логики от элементов интерфейса был использо- ван паттерн ViewModel

```
public class TimerViewModel extends ViewModel {
        MutableLiveData<TimeContainer> currentTimeOnClock = new
        MutableLiveData<>();
        MutableLiveData<Timer> currentWorkingTimer = new
    MutableLiveData<>(); MutableLiveData<List<TimerData>>
currentHistory = new
    MutableLiveData<>();
    MutableLiveData<TimerControlledButtonGroup.State>
buttonsState = new MutableLiveData<>();
public TimerViewModel() {
```

```
        currentTimeOnClock.setValue(new TimeContainer(0));
    } ...
```

Взаимодействие между UI элементами и различными потоками происходит с помощью реактивных оберток над данными.

ЗАКЛЮЧЕНИЕ

Изученные методы и подходы разработки показали свою эффективность на практике. За счет внедрения Agile методологии в разработку программного продукта, улучшилось взаимодействие внутри команды, выявились приоритеты разработки среди возможных путей решения на каждом этапе. В результате был на- писан таймер на языке Java, работающий под Android.

Цели и задачи практики были выполнены, получен ценный опыт работы с компанией.

Ссылка на исходный код: https://github.com/marilisok/timer-android.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Android Developers: https://developer.android.com/

2.ViewModelOverView:https://developer.android.com/topic/libraries/architecture/viewmodel

3. Dagger: https://dagger.dev/