**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В. Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Курсовая работа  
по дисциплине: «Тестирование программных систем»  
тема: «Автоматизация UI-тестирования веб-приложения «Телеграф»»

Выполнила: ст. группы ПВ-41  
Цилюрик М.А.  
Проверил: Поляков В.К.

Белгород 2020

Оглавление

[Введение 3](#_Toc61660904)

[Тестирование пользовательского интерфейса 3](#_Toc61660905)

[Инструмент для автоматизации действий браузера Selenium 3](#_Toc61660906)

[Установка инструментов тестирования 3](#_Toc61660907)

[Тестирование веб-приложения «Телеграф» 5](#_Toc61660908)

[Тестирование регистрации(RegisterTest) 5](#_Toc61660909)

[Тестирование авторизации (LoginTest) 7](#_Toc61660910)

[Тестирование домашней страницы (HomeTest) 10](#_Toc61660911)

[Тестирование всего сайта (main) 12](#_Toc61660912)

[Исходный код 13](#_Toc61660913)

[Заключение 13](#_Toc61660914)

# **Введение**

В ходе выполнения данной работы будет реализовано интеграционное тестирование веб-приложения. Каждое программное обеспечение требует тестирования. Тестирование графического интерфейса позволяет: выявить ошибки функциональности, выявить потерю данных, передаваемых через элементы интерфейса, найти ошибку в интерфейсе. Также, тестирование должно отвечать требованиям к реакции системы на пользовательский ввод, на время отклика на команды пользователя. Функциональное тестирование UI содержит в себе: анализ требований к интерфейсу, разработки тест требований, выполнение тестовых примеров, составление отчетов о проблемах.

# **Тестирование пользовательского интерфейса**

Графический интерфейс пользователя (Graphical user interface, GUI) – разновидность пользовательского интерфейса, в котором элементы интерфейса (меню, кнопки, значки, списки и т. п.), представленные пользователю на дисплее, исполнены в виде графических изображений.

Задачей тестирования графического интерфейса пользователя является обнаружение ошибок следующего характера:

* Ошибки в функциональности посредством интерфейса
* Необработанные исключения при взаимодействии с интерфейсом
* Потеря или искажение данных, передаваемых через элементы интерфейса
* Ошибки в интерфейсе (несоответствие проектной документации, отсутствие элементов интерфейса)

# **Инструмент для автоматизации действий браузера Selenium**

Selenium – это инструмент для автоматизированного управления браузерами.

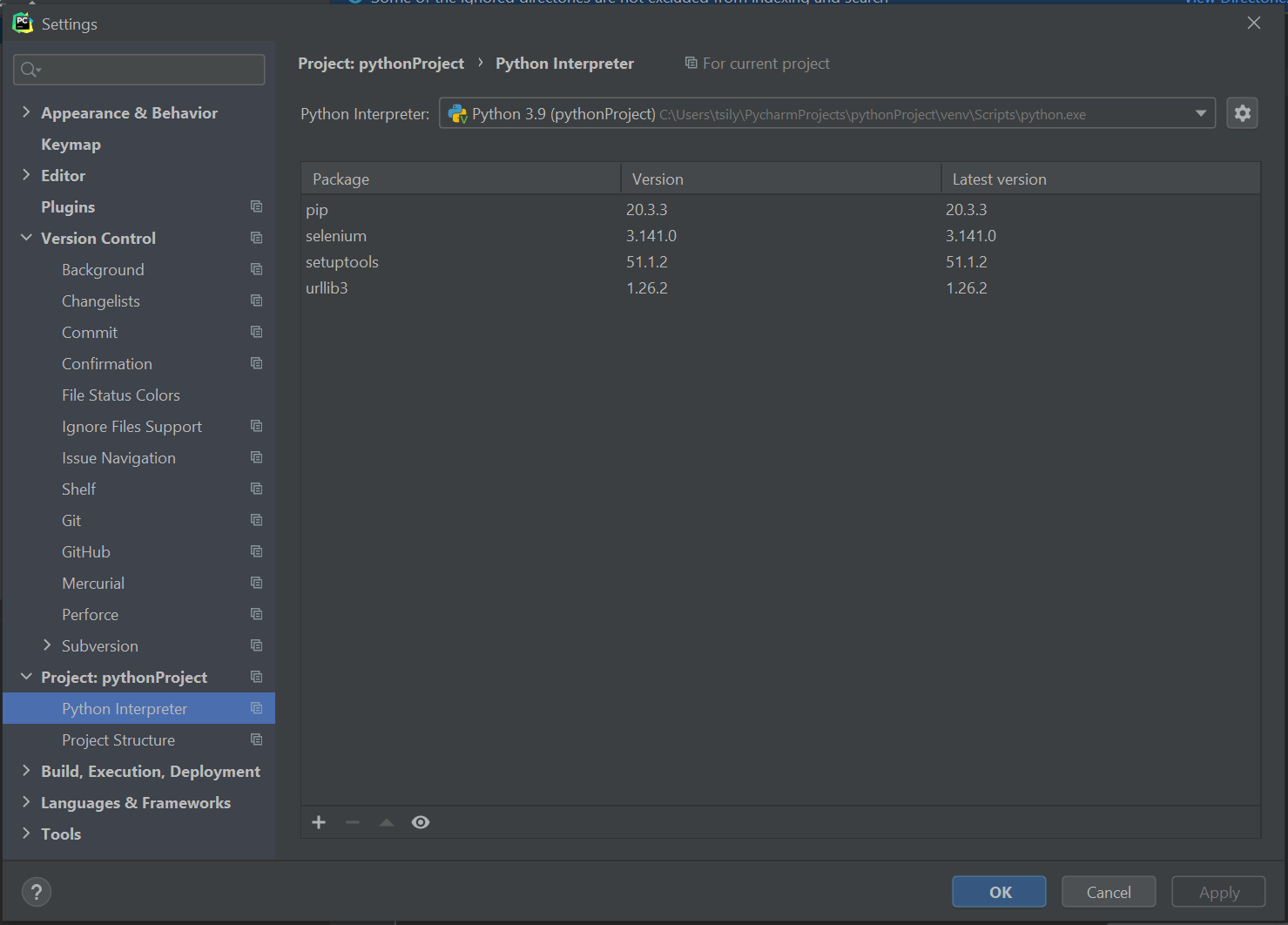
Чаще всего Selenium используется для автоматизации тестирования веб-приложений. Однако при помощи Selenium можно автоматизировать любые рутинные действия, выполняемые через браузер.

Разработку Selenium поддерживают производители браузеров. Они адаптируют браузеры для более тесной интеграции с Selenium, а иногда даже реализуют в браузере встроенную поддержку Selenium.

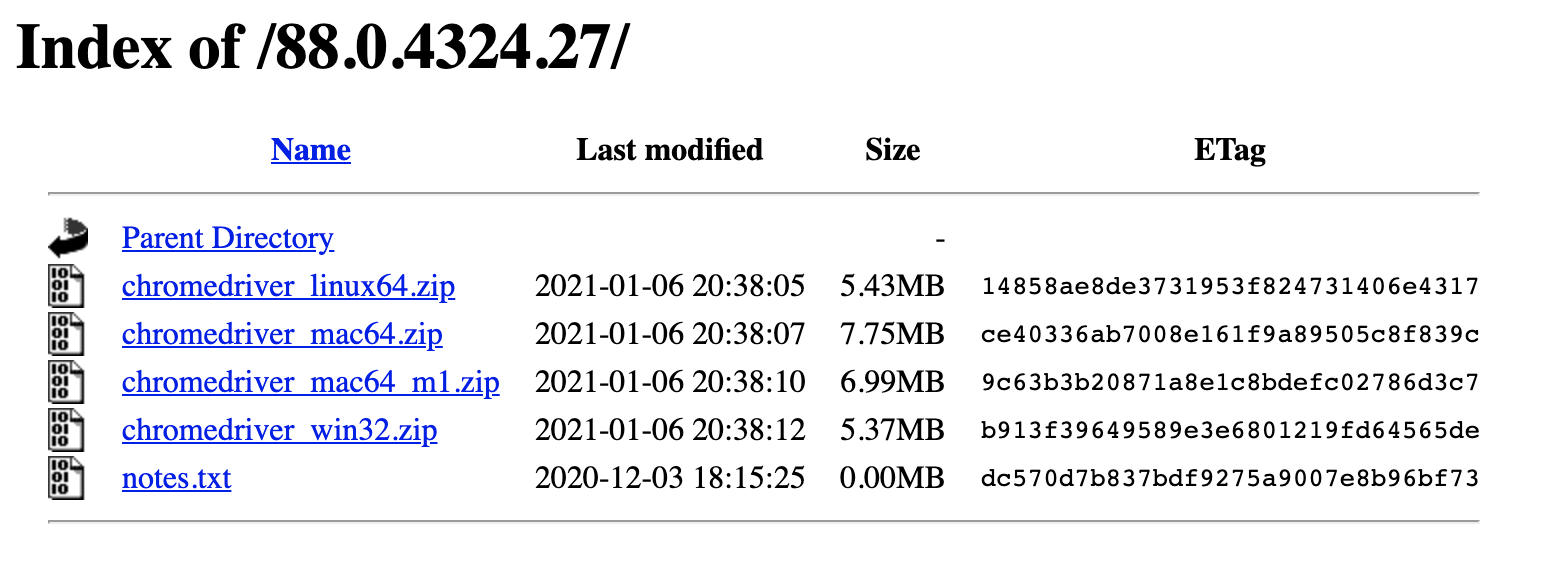
Selenium позволяет писать сценарии практически на любом языке программирования. Он является ключевым компонентом множества открытых и проприетарных инструментов автоматизации. Selenium позволяет управлять браузером удаленно, благодаря чему можно создавать распределённые стенды, состоящие из множества машин с разными операционными системами и браузерами, и даже запускать браузеры в облаках.

# **Установка инструментов тестирования**

Для реализации тестов был выбран язык программирования Python и библиотека selenium. Для использования selenium нужно установить соответствующую библиотеку в настройках проекта:



В качестве браузера, который будет запускать тесты выбран Google Chrome. Для использования в Selenium chrome необходимо установить web-драйвер. С официального сайта chrome качаем драйвер для используемой системы -Windows 32 бита.



Также в проекте была импортирована библиотека unittest для написания тестов.

Unittest -стандартный модуль для написания юнит-тестов на Python, который имеет несколько полезных функций:

* можно собирать тесты в группы
* собирать результаты выполнения тестов
* ООП стиль позволяет уменьшить дублирование кода при схожих объектах тестирования

Далее в классах для тестирования будет использована следующая конструкция для применения установленных инструментов (пример одного из далее реализованных классов):

def setUp(self):  
 chromePath = "C:/Users/tsily/PycharmProjects/pythonProject/chromedriver.exe"  
 self.driver = webdriver.Chrome(executable\_path=chromePath)  
 self.driver.get("http://localhost:8081/")  
 time.sleep(1)

В переменной chromePath должен быть указан путь до скачанного раннее драйвера chromedriver, который затем передается в метод webdriver.Chrome для взаимодействия с браузером. Также в методе driver.get указываем ссылку на начальную страницу, которая нужна для тестов данного класса. Методы setup и teardown реализованы подобным образом в каждом классе тестов.

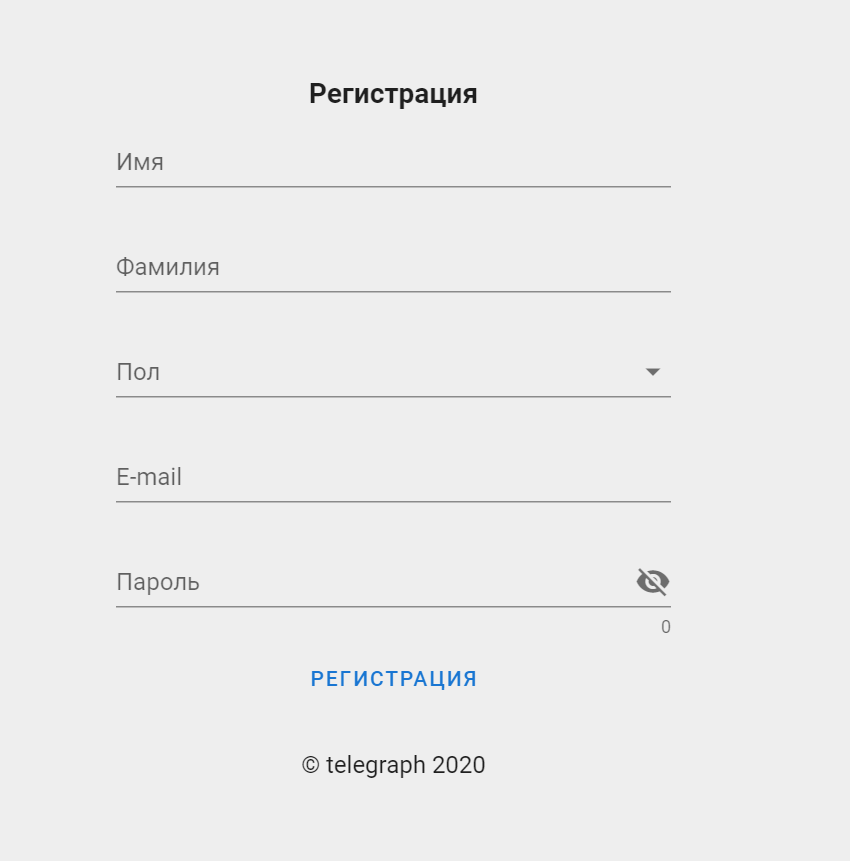
# **Тестирование веб-приложения «Телеграф»**

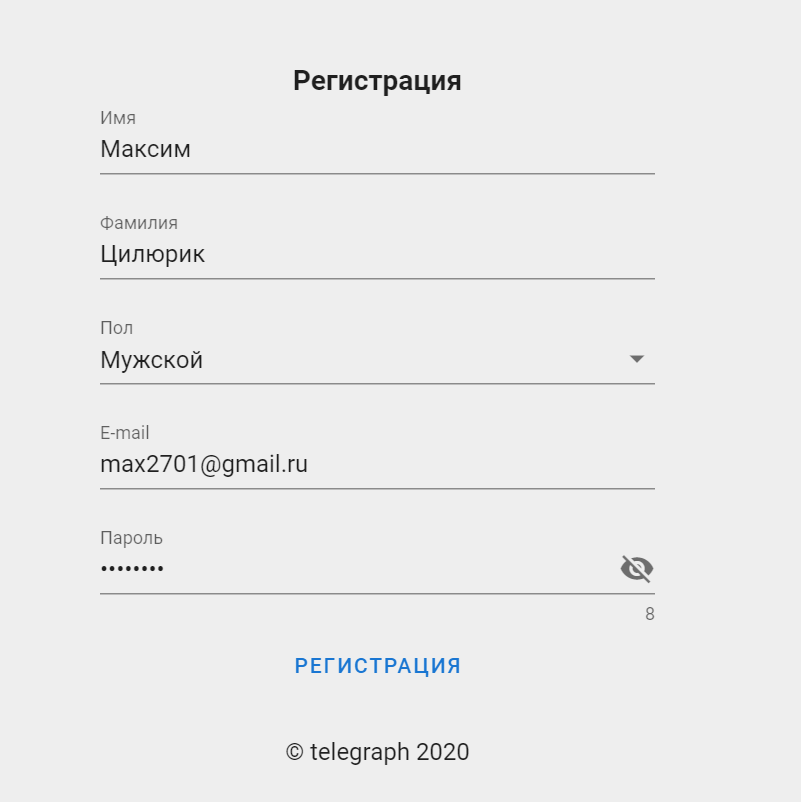
Для тестирования веб-приложения с информацией о фильмах были реализованы отдельные тесты для компонентов, связанных с главной страницей, с пользователями (профиль, регистрация и авторизация)

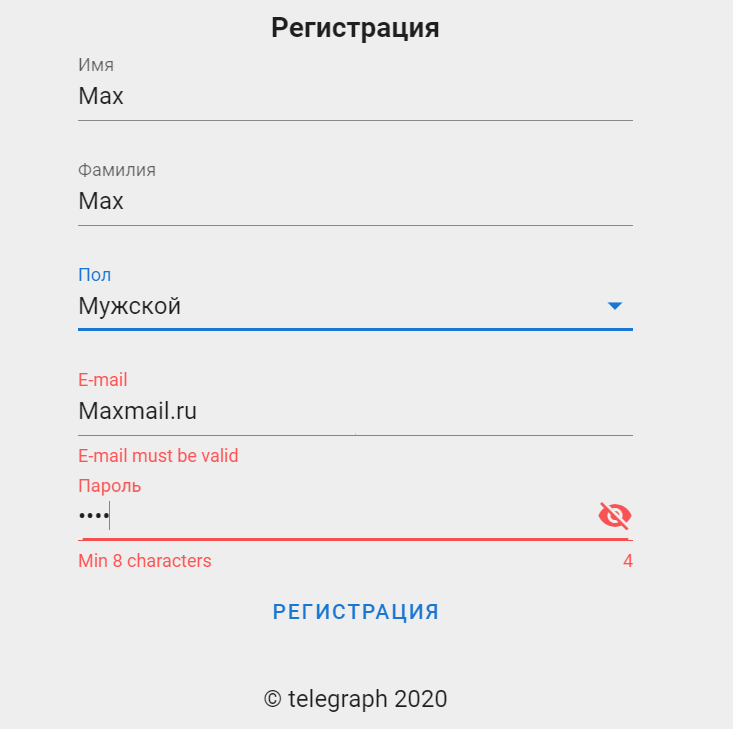
Для запуска всех тестов вместе они также были вынесены в отдельный файл main.py проекта.

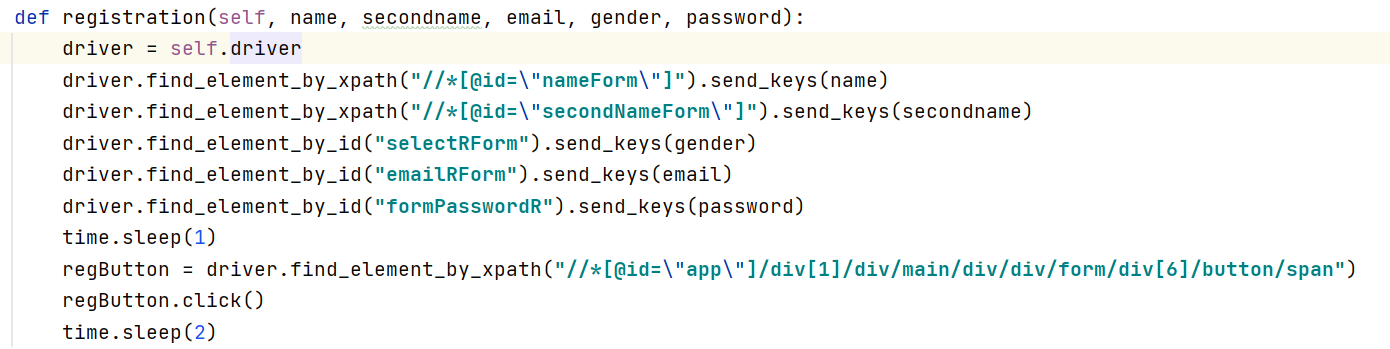
## **Тестирование регистрации(RegisterTest)**

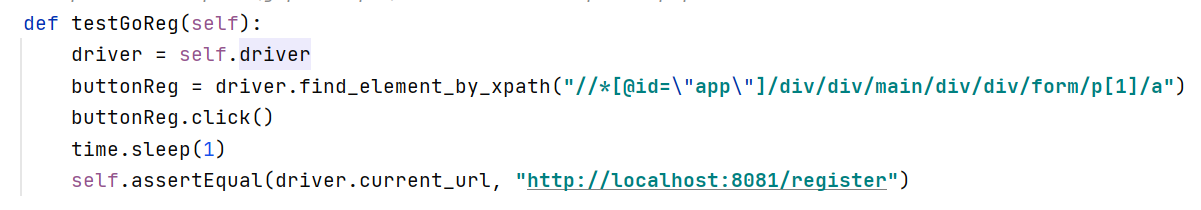
В тестах для данной страницы сначала была реализована вспомогательная функция регистрации пользователя, которая с помощью метода send\_keys заполняет текстовые поля формы регистрации соответствующими значениями, переданными как аргументы:

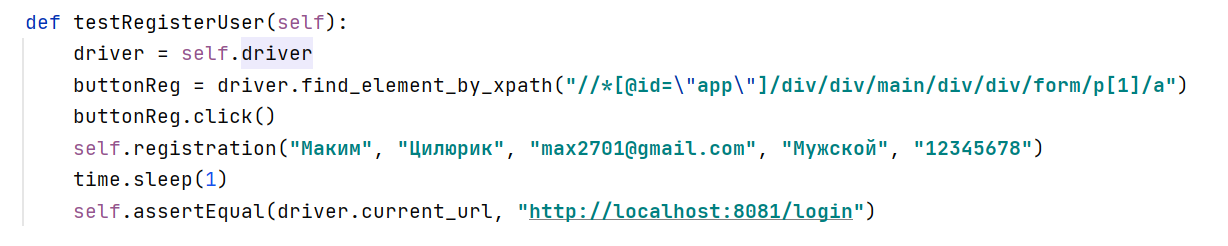


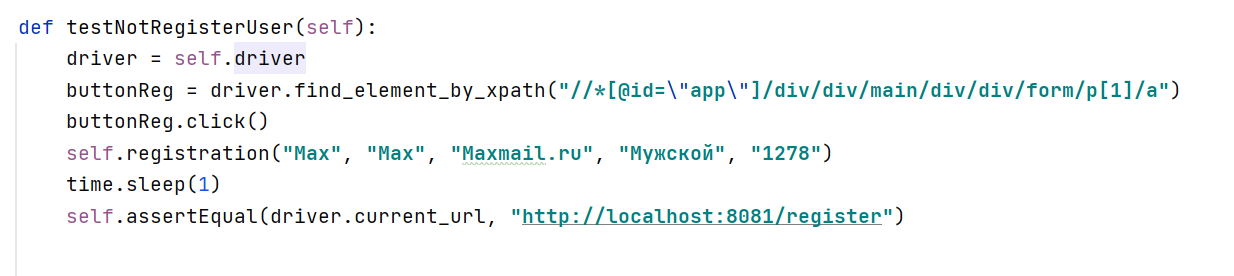




Вспомогательная функция для заполнения полей формы.

1. Тестирование перехода из страницы авторизации на страницу регистрации
2. Тестирование регистрации нового пользователя. В случае удачной регистрации произойдет переход на страницу авторизации.

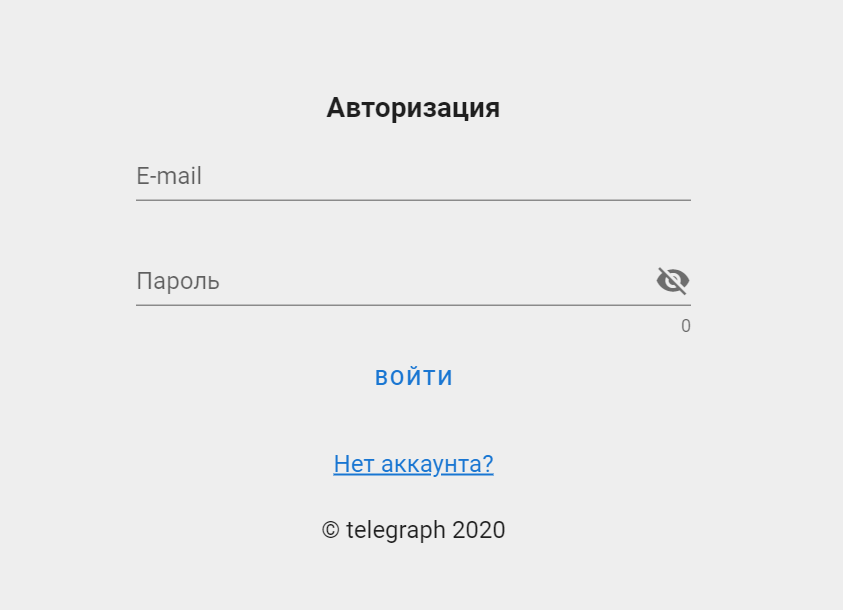


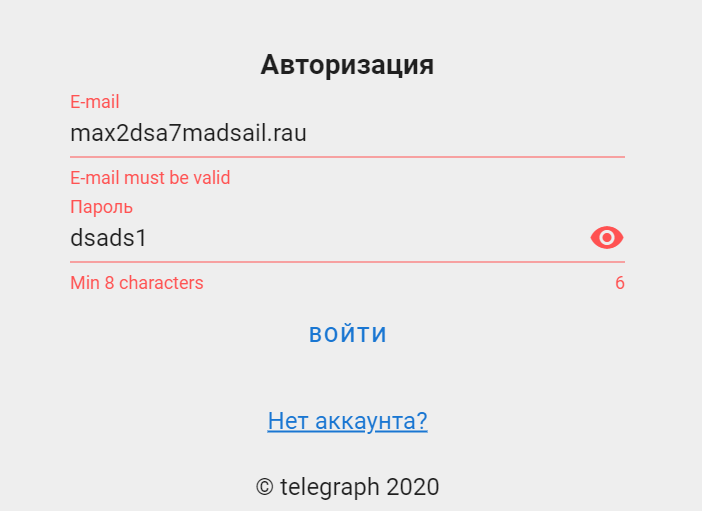
1. Тестирование валидации формы. В случае если пользователь неверно заполнит поля, при нажатии на кнопку, ничего не произойдет.

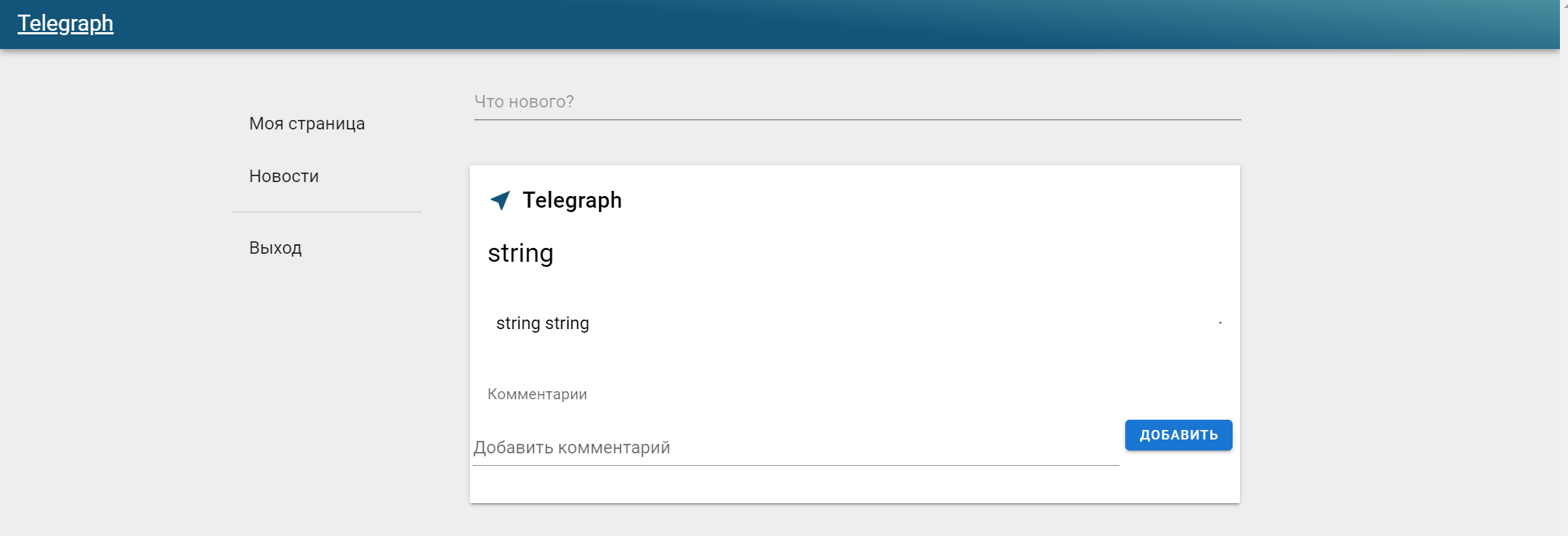
## **Тестирование авторизации (LoginTest)**

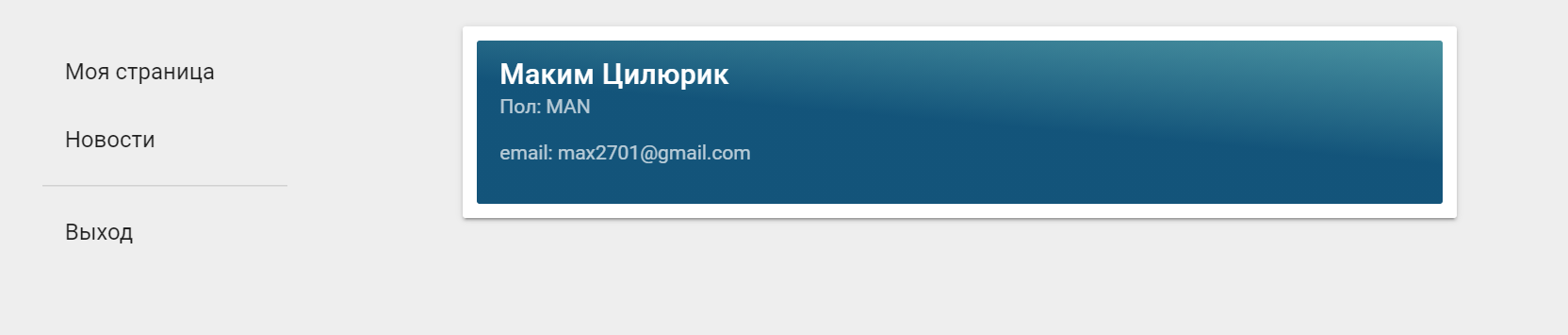
Для авторизации необходимо проверить следующие ситуации. При успешной авторизации пользователь попадает на начальную страницу(главную), где может перейти в профиль, вернуться на начальную страницу и создать новый пост и к нему комментарий (будет протестировано в другом модуле). Из этого следует, что необходимо выделить следующие тест-кейсы для авторизации:

1. Удачная авторизация и переход на главную страницу.
2. Авторизация с ошибочными данными
3. Авторизация с невалдным логином
4. Авторизация с невалидным паролем
5. Авторизация с переходом в профиль и проверкой данных в профиле.





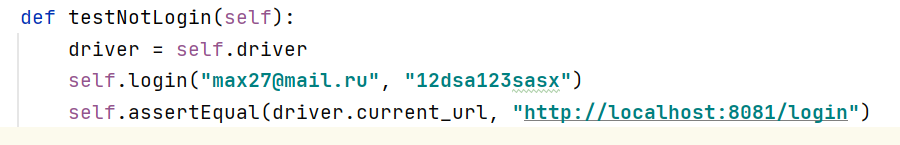




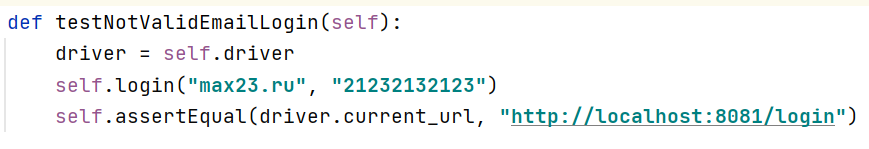
1. Удачная авторизация и переход на главную страницу.



1. Авторизация с ошибочными данными. В данном тест кейсе при ошибочных данных пользователь останется на странице авторизации.



1. Авторизация с невалдным логином. Данном случае пользователь не сможет авторизоваться, так как поле login имеет невалидные данные.



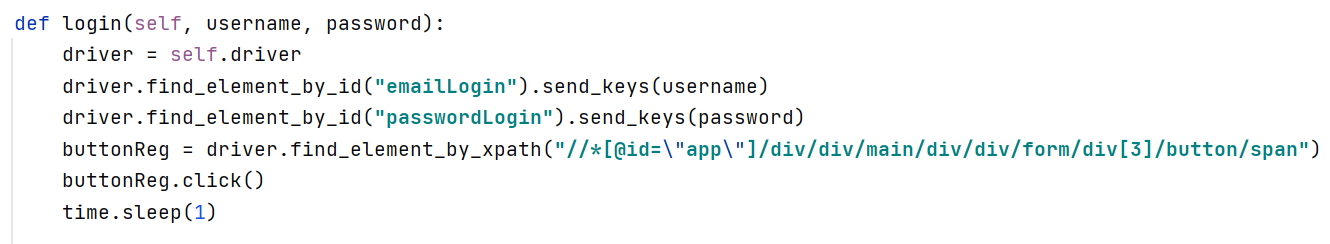
1. Авторизация с невалдным паролем. Данном случае пользователь не сможет авторизоваться, так как поле password имеет невалидные данные.



1. Успешная авторизация с переходом на страницу профиля и проверкой данных авторизованного пользователя.



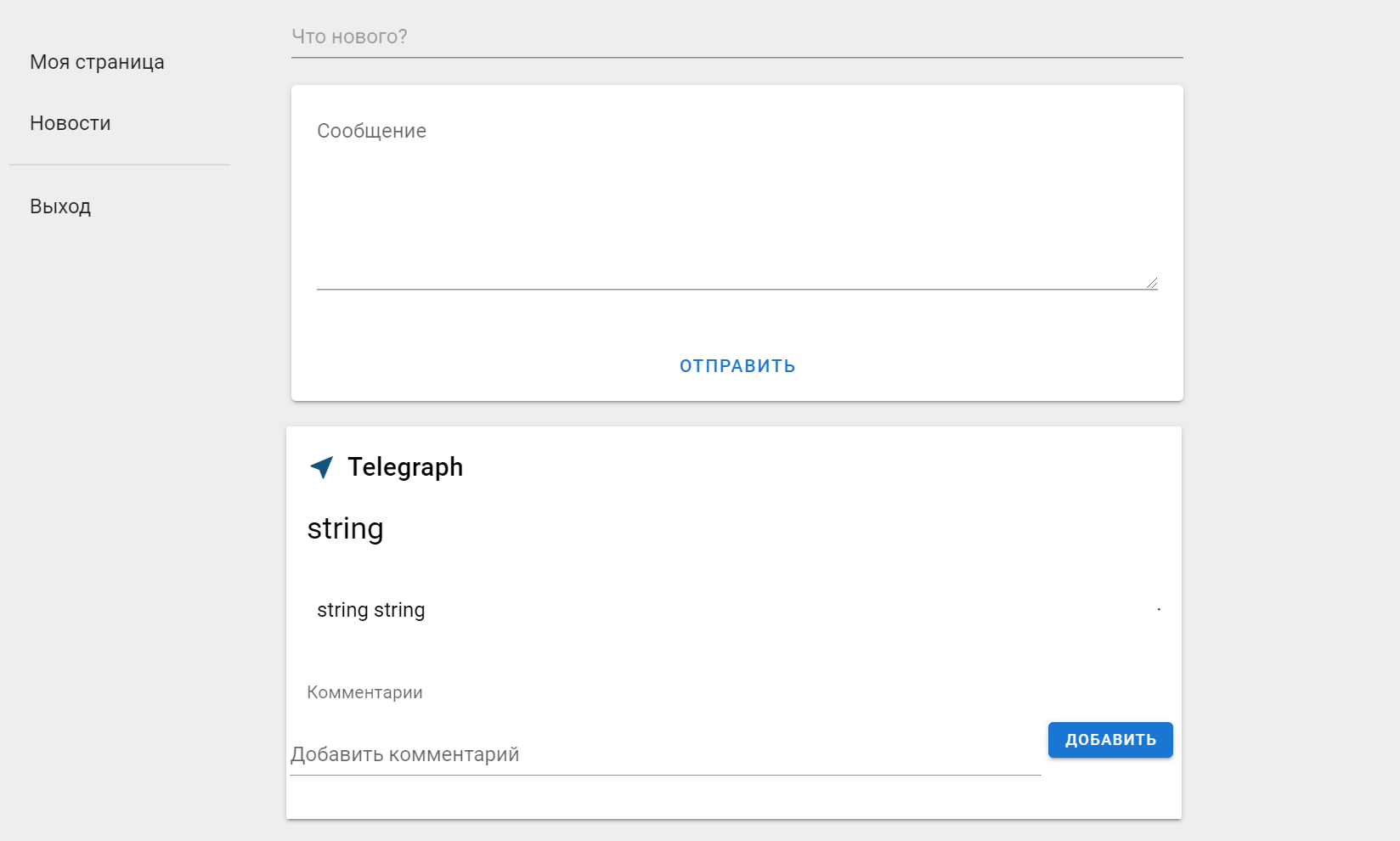
Так же была объявлена вспомогательная функция для заполнения полей формы авторизации.

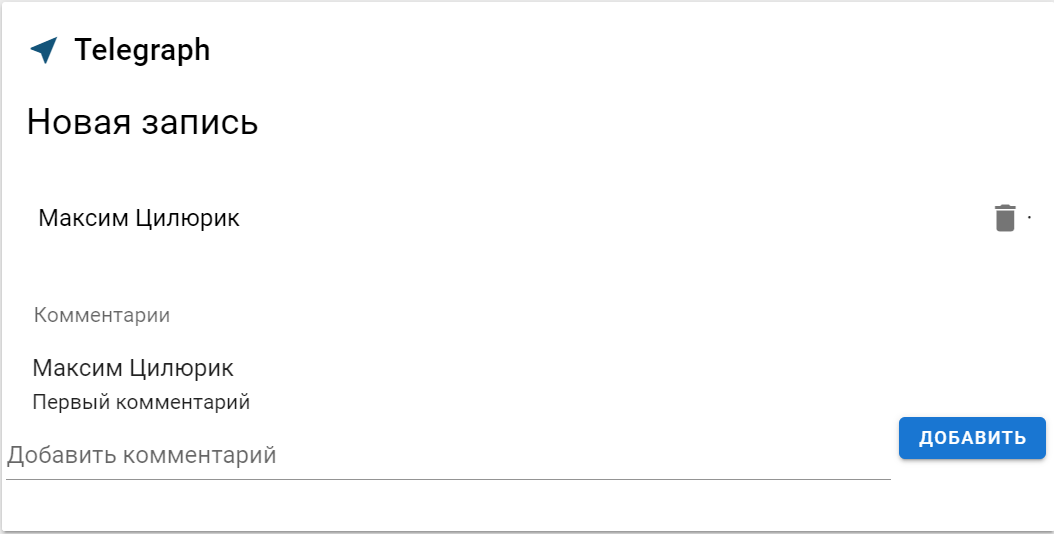


## **Тестирование домашней страницы (HomeTest)**

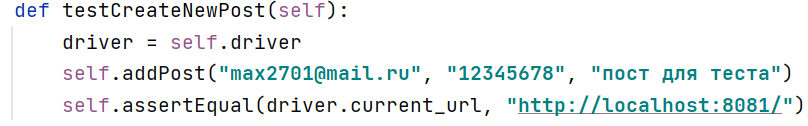
Домашняя страница представляет собой список постов, форму для создания поста, так боковую навигационную панель для перехода на свою страницу, а так же для выхода из приложения. Были выделены следующие ситуации для тестирования:

1. Создание формы поста
2. Удаление созданного поста
3. Добавление комментария

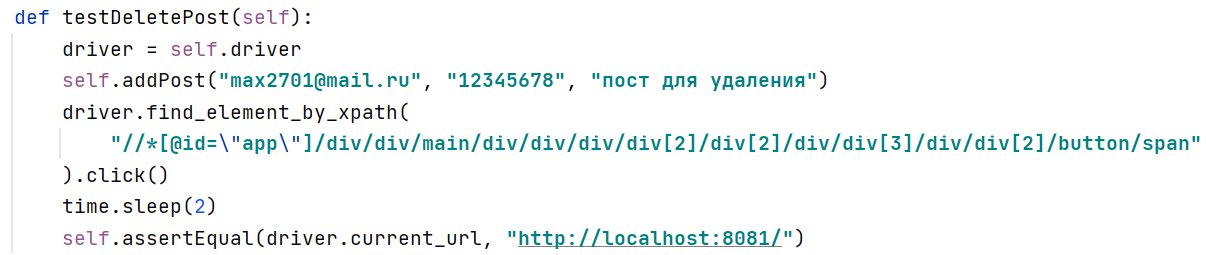




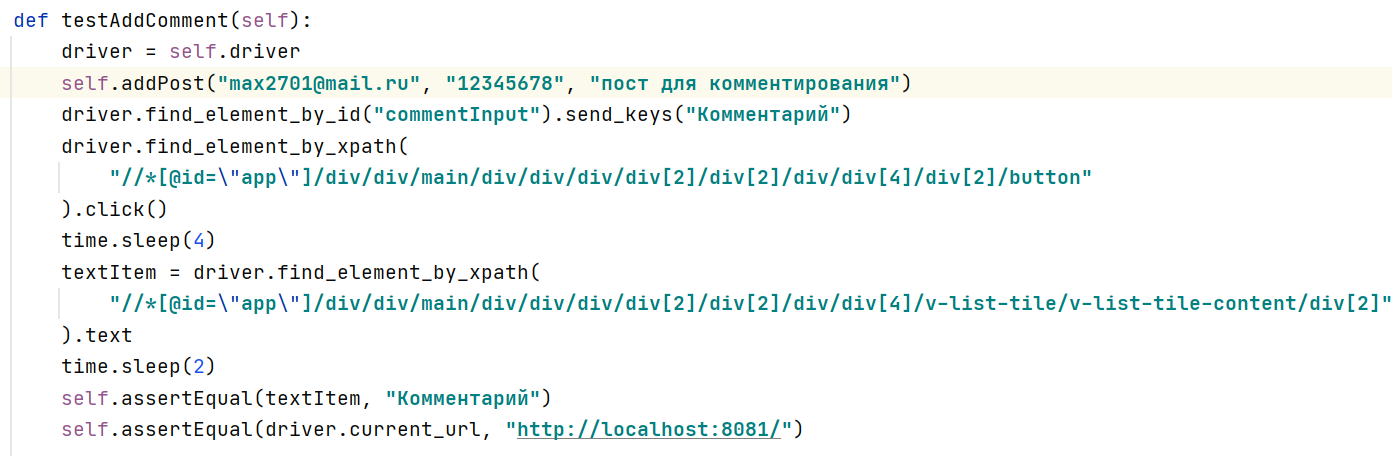
1. Тестирование создания нового поста. При создании нового поста он отображается на главной странице. Для это необходимо нажать на поле ввода «что нового?», где появится форма полем ввода и кнопкой отправить. При этом должен отработать автофокус на поле ввода.



1. Тестирования удаления происходит следующим образом. Авторизованный пользователь создает пост, и следом его удаляет, нажав на кнопку «корзины».



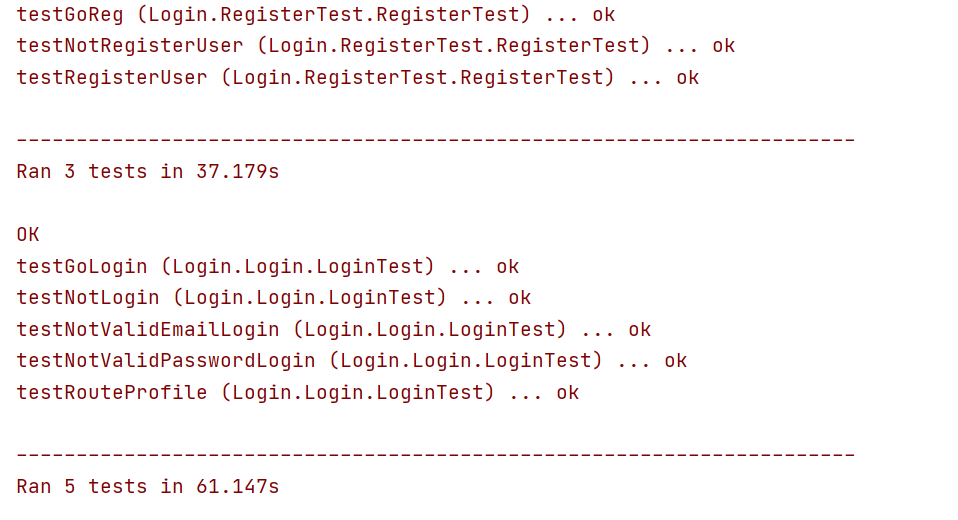
1. Добавления комментария. Авторизованный пользователь создает пост, а также создает добавляет к нему комментарий.

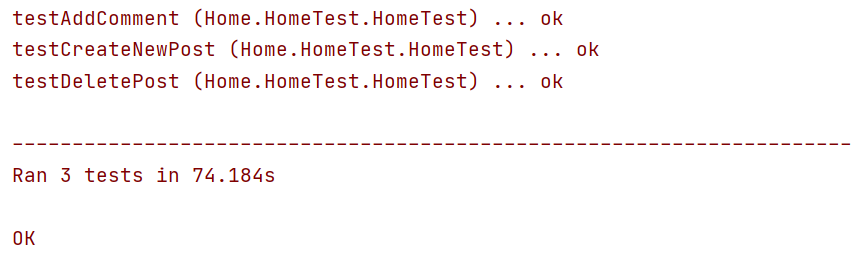


## **Тестирование всего сайта (main)**

Для запуска всех тестов сразу они были объединены в файле main.py.

import unittest  
from Home.HomeTest import HomeTest  
from Login.RegisterTest import RegisterTest  
from Login.Login import LoginTest  
  
  
def runTests(testClasses):  
 loader = unittest.TestLoader()  
 suitesList = []  
 for testClass in testClasses:  
 suite = loader.loadTestsFromTestCase(testClass)  
 suitesList.append(suite)  
 runner = unittest.TextTestRunner(verbosity=2)  
 runner.run(unittest.TestSuite(suitesList))  
  
  
def main():  
 runTests([RegisterTest])  
 runTests([LoginTest])  
 runTests([HomeTest])  
  
  
if \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()





## **Исходный код**

Исходный код доступен по ссылке: https://github.com/MaxTsilyurik/testing\_curs

# **Заключение**

В ходе выполнения курсовой работы было реализовано автоматизированное тестирование веб-приложения. Ознакомились с средством автоматизации действий веб-браузера Selenium. Была установлена связка Python + Selenium + Chrome Web Driver для тестирования веб-приложения «КиноПолка». Написаны тесты для отдельных веб-страниц, их различных компонентов, протестированы переходы на различные страницы через ссылки. Также протестированы такие элементы, как кнопки, поля ввода и т.д. Протестировали написанными тестами реализованный проект и получили результаты.