Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

АВТОМАТИЗАЦИЯ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ НА ПРИМЕРЕ САЙТА QUEST.RUN

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Лопатникова Павла Юрьевича

Научный руководитель
доцент, к. ф.-м. н. _____ И. А. Батраева

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2016

ВВЕДЕНИЕ

Актуальность темы. В настоящее время широко распространены концепции разработки программного обеспечения (ПО) с использованием активного взаимодействия между заказчиком и командой разработки в процессе реализации и развития ПО. Многие компании используют гибкие методологии разработки, предполагающие постоянные изменения продукта, в связи с этим автоматизация процесса тестирования часто становится необходимостью, поскольку существенно экономит ресурсы при правильном подходе и использовании. В выпускной квалификационной работе (ВКР) автоматизация тестирования будет рассмотрена на примере сайта Quest.run, который был разработан как платформа для людей, у которых есть квест (интеллектуальная игра), но нет площадки для управления им. **Цели**. Целью работы является рассмотрение подходов к автоматизированному функциональному тестированию, его достоинств и недостатков, а также проектирование и реализация покрытия основного функционала веб-сайта Quest.run автотестами с помощью языка программирования Python и его модуля Unittest, используя программную библиотеку для управления браузерами Selenium WebDriver. **Задачи**. В выпускной квалификационной работе работе ставятся задачи:

- рассмотреть достоинства и недостатки автоматизации функционального тестирования;
- рассмотреть подходы к автоматизации функционального тестирования;
- выделить функционал веб-сайта Quest.run, который необходимо и можно подвергать автоматизированному тестированию;
- разработать и реализовать автоматизированные тесты с использованием модуля Unittest языка Python и Selenium WebDriver.

Общая характеристика материала исследования. Автоматизированное тестирование не всегда может быть полезным. В работе рассмотрены плюсы и минусы автоматизации тестирования. К автоматизации тестирования существует несколько подходов. Можно подвергать автотестам непосредственно код программы, а можно тестировать использую GUI (графический интерфейс пользователя) или API(программный интерфейс, который предоставляет продукт). В данной работе рассмотрены вышеописанные подходы, а также описана реализация автоматизации тестирования на примере сайта Quest.run с использованием GUI. **Структура ВКР**. ВКР содержит три главы. Первая глава

называется «Тестирование программного обеспечения». В ней приведены основные определения, связанные с работой, рассмотрены виды тестирования, а также отдельно рассмотрено автоматизированное тестирование, его достоинства и недостатки и подходы к нему. Вторая глава называется «Quest.run». В ней описывается сайт, для которого разработаны и реализованы автоматизированные тесты. Третья глава называется «Автоматизация тестирования сайта Quest.run». В ней описан функционал, для которого разработаны автотесты, инструменты, которые были использованы в ходе реализации автотестов, а также подробно рассмотрены реализованные автотесты вместе с кодом и приведены скриншоты. [1–10]

1 Основное содержание работы

Был выбран функционал сайта Quest.run, который должен быть покрыт автотестами:

1. Авторизация зарегистрированного ранее пользователя.
2. Регистрация и авторизация свежезарегистрированного пользователя.
3. Невозможность регистрации с email, который уже зарегистрирован.
4. Создание квеста ранее зарегистрированным пользователем.
5. Создание квеста свежезарегистрированным пользователем.
6. Создание шаблона расписания для квеста.
7. Создание слота в шаблоне расписания для квеста.
8. Назначение шаблона расписания на определенную дату.
9. Бронирование свободного слота квеста.
10. Составление отчёта для завершённого слота.
11. Добавление нового пользователя, используя профиль ранее зарегистрированного пользователя.
12. Добавление нового пользователя, используя профиль свежезарегистрированного пользователя.

Инструментарий. Сайт Quest.run написан на языке Python. Поэтому в качестве языка для автоматизации тестирования был выбран Python. В качестве среды разработки была выбрана «JetBrains PyCharm Community Edition 5.0.5». Модуль Unittest входит в стандартную библиотеку Python и служит базовым инструментом для организации функциональных и регрессионных тестов.Selenium WebDriver по назначению представляет собой драйвер браузера, то есть программную библиотеку, которая позволяет разрабатывать программы, управляющие поведением браузера. **Подготовительный этап реализации автотестов**. Тесты будут производиться на тестовом стенде, который полностью эмулирует реальный сайт и имеет такое же окружение. Стенд разворачивается на localhost. Для упрощения тестирования перед началом тестов создается база данных, в который используется один пользователь с почтой «test@test.com» и паролем «test». У него в проекте имеются два квеста: «first quest» и «second quest». Для первого добавлены несколько слотов. В результате стартовая страница имеет вид (см. рисунок 1). **Тест авторизации пользователя, который уже есть в базе**. Делаем проверку на то, что в заголовке содержится правильное название, т.е. убеждаемся, что попали по нужному

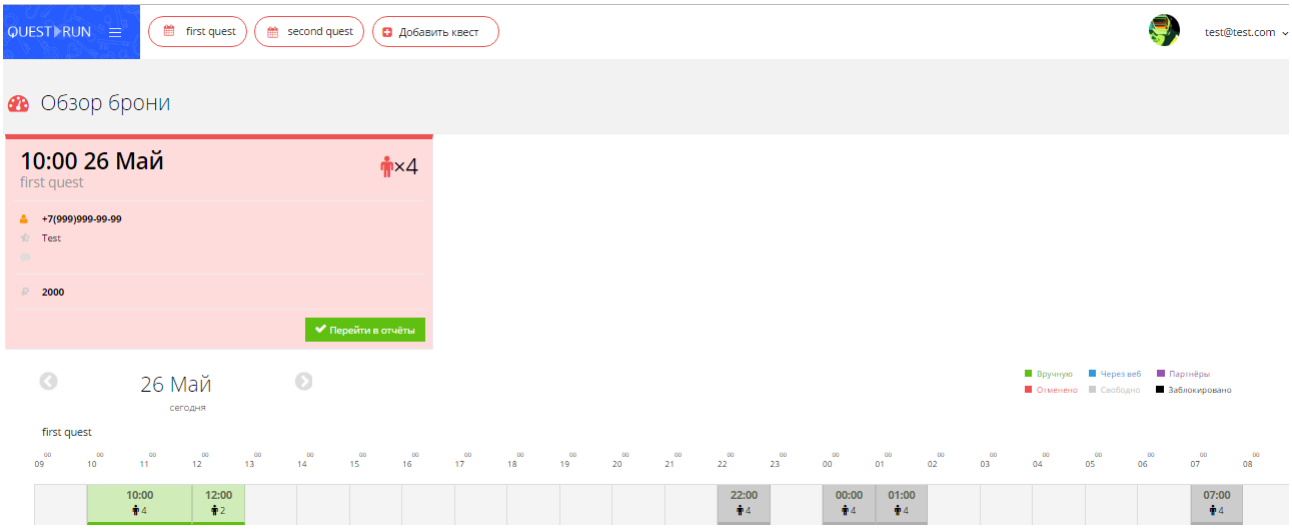

Рисунок 1 – Стартовая страница Quest.run

адресу. Затем используются стандартные методы, который предоставляются webdriver. Ищем элементы по имени и вводим в них данные с помощью метода $send\_keys()$. Далее находим кнопку входа по имени класса, нажимаем ее, используя стандартный метод webdriver $click()$. В результате должна произойти авторизация. Чтобы в этом убедиться, мы находим нужный элемент по CSS-селектору, в котором содержится почта пользователя и сравниваем его с переданным в тест параметром email. Форма авторизации до нажатия кнопки входа на рисунке 2. **Тест регистрации пользователя с незанятым email**. Убеждаемся, что попали по адресу и нажимаем кнопку создания аккаунта. Далее генерируем случайное восьмибитное число. Это необходимо для придания email уникальности. Затем вводим необходимые для регистрации данные и нажимаем кнопку регистрации. Форма регистрации до нажатия кнопки регистрации выглядит на рисунке 2. Для того, чтобы убедиться, что регистрация прошла успешно выполняем авторизацию с введенными при регистрации данными. Для этого вызываем тест авторизации и передаем в него сгенерированную почту. В результате выполнения теста должна произойти успешная авторизация нового пользователя. **Тест регистрации пользователя с занятым email**. Убеждаемся, что попали по адресу и нажимаем кнопку создания аккаунта. Теперь вводим в качестве email нового пользователя уже использующийся. Нажимаем кнопку регистрации. В качестве валидации используем поиск элемента по id. Данный элемент это ошибка, которая говорит о том, что email уже занят. **Тест возможности создания квеста ранее зарегистрированным пользователем**. Вызываем тест авторизации и запоминаем имя

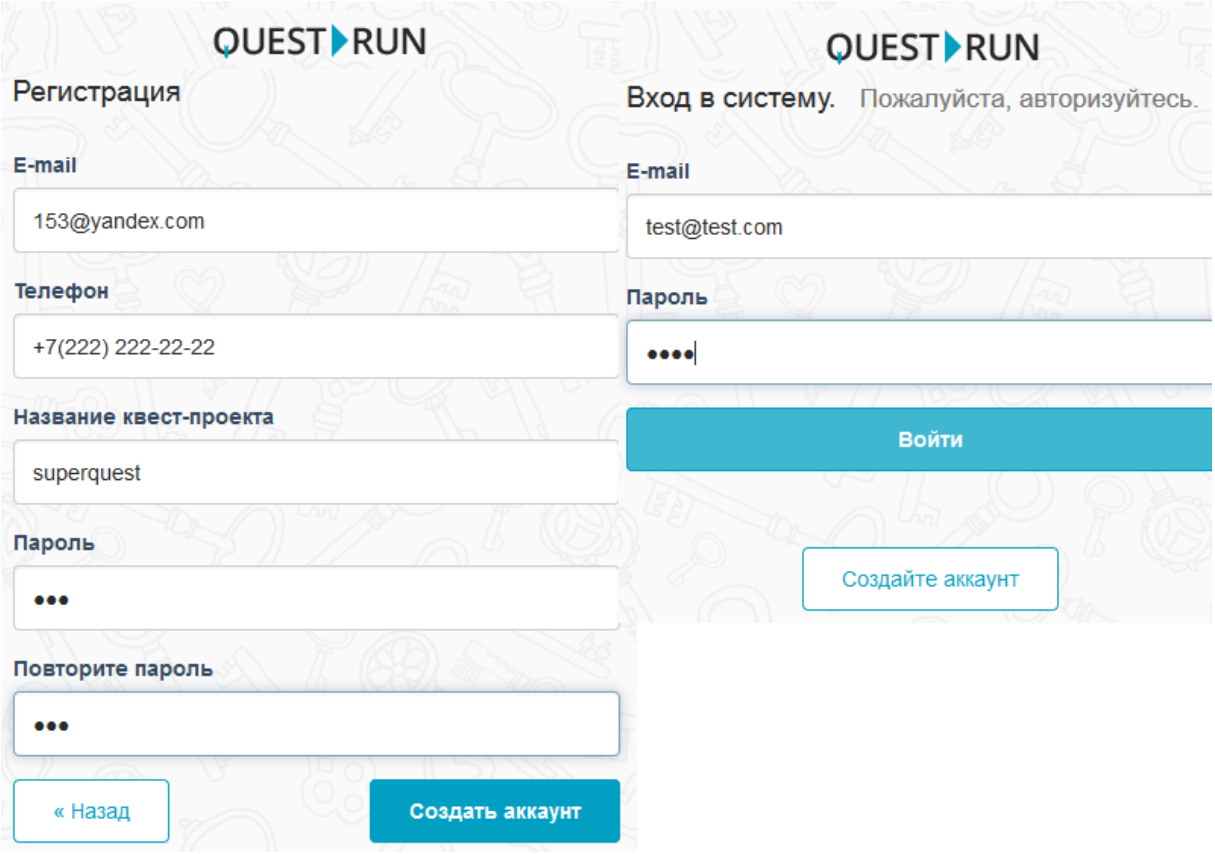

Рисунок 2 – Заполненные формы авторизации и регистрации

добавляемого квеста. Теперь нажимаем кнопку создания квеста, вводим имя квеста и его короткое имя, нажимаем кнопку добавления квеста. Осталось убедиться, что новый квест добавился. Для этого открываем список имеющихся квестов и ищем, есть ли среди имеющихся добавленный нами квест. Результат успешно выполненного теста на рисунке 3. **Тест возможности создания квеста свежезарегистрированным пользователем**. Вызываем тест регистрации нового пользователя и запоминаем имя добавляемого квеста. Нажимаем кнопку создания квеста, вводим имя квеста и его короткое имя, нажимаем кнопку добавления квеста. Чтобы убедиться, что новый квест добавился достаточно найти ссылку на странице с его именем. Выполнять действий как в предыдущем тесте незачем, т.к. у этого пользователя еще нет квестов. **Тест возможности создания шаблона расписания**. Вызываем тест авторизации, запоминаем имя добавляемого расписания и переходим на страницу расписания. Нажимаем кнопку добавления расписания, вводим его имя, добавляем расписание. Убеждаемся в том, что расписание создалось. Для этого ищем второе расписание, т.к. первое уже было в базе и проверяем, что его имя такое же, с каким мы создали расписание. Результат успешно выполненного теста

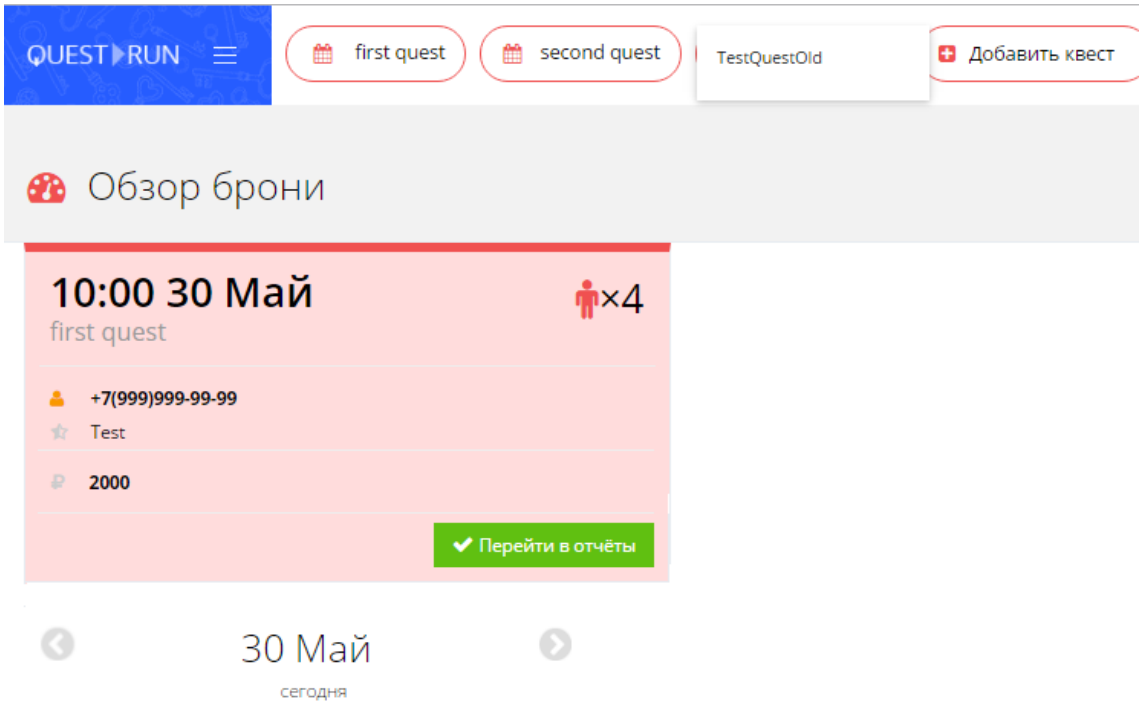

Рисунок 3 – Добавленный квест для имеющегося в базе пользователя

на рисунке 4. **Тест возможности создания слота в шаблоне расписания**. Вызываем тест авторизации и переходим на страницу расписания. Обращаемся к свободному слоту в шаблоне расписания, нажимая на него, вводим необходимую информацию. Указываем минимальное, среднее, максимальное количество игроков, цену за слот, цену за превышения количества человек и время окончания слота. Добавляем его.Теперь открываем этот же слот и соотносим введенные данные с имеющимися. В результате успешного завершения теста в расписание будет добавлен слот 4. **Тест возможности назначить шаблон расписания на определенный день**. Вызываем тест авторизации и переходим на страницу расписания. Запоминаем элемент расписания, который уже имеется. Далее узнаем сегодняшнюю дату, прибавляем один день. Теперь находим этот день на линейке. Затем применяем функцию $drag - and - drop$ для этих элементов, таким образом назначая шаблон на выбранный день. Назначенный на следующий день шаблон на рисунке 5. Переходим на главную страницу. Перемещаем дату календаря на день вперед. Теперь убеждаемся в том, что на главной странице на текущую дату в календаре назначен слот на 10 часов, который присутствует в назначенном нами шаблоне расписания. Результат успешно выполненного теста на рисунке 6. Слоты шаблона присутствуют на линейке на главной странице. **Тест возможности бронирования свободного слота**. Вызываем тест авторизации, открываем заранее определен-

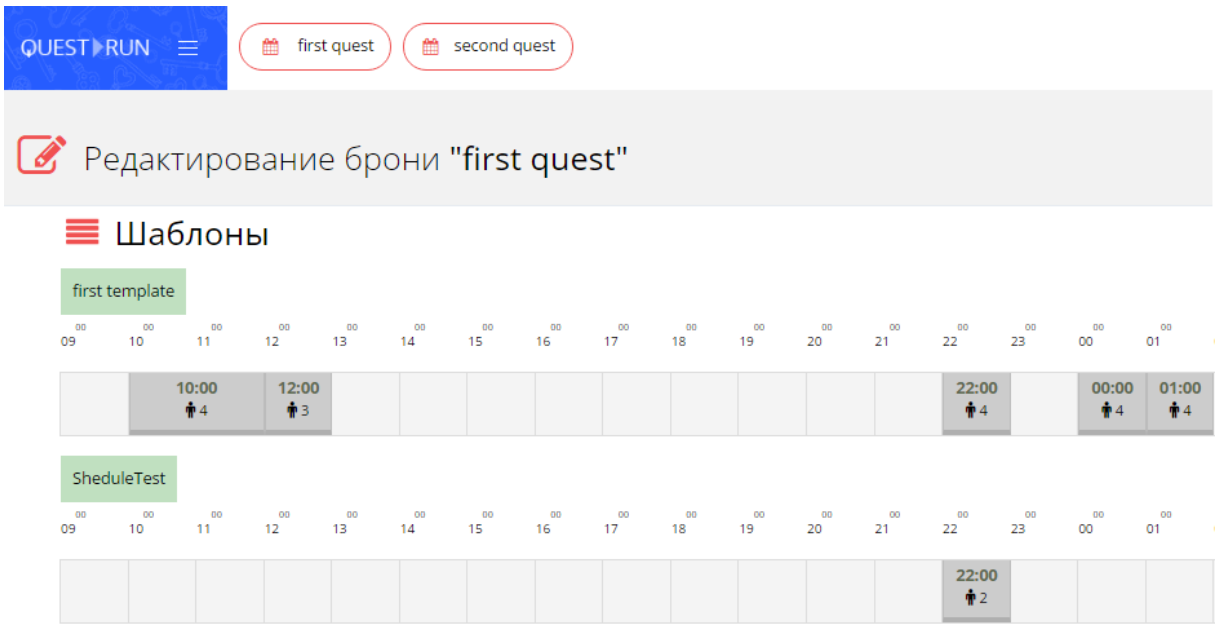

Рисунок 4 – Добавлен шаблон SheduleTest и слот в нём

Рисунок 5 – Добавлен шаблон расписания на следующий день

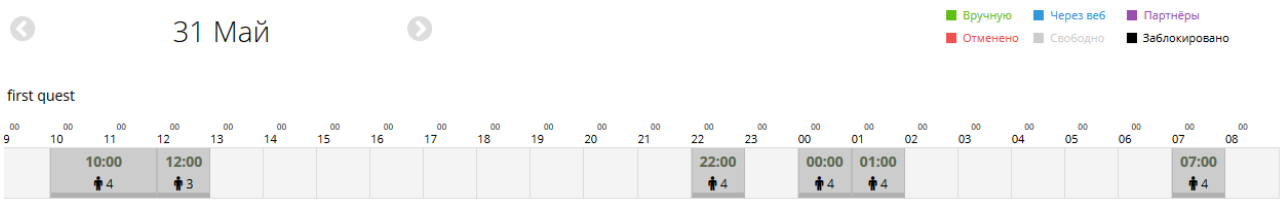

Рисунок 6 – Слоты назначенного шаблона присутствуют на главной странице

ный свободный слот на линейке на главной странице. Вводим необходимые данные: телефон, имя, количество игроков. Нажимаем кнопку бронирования слота. Обновляем главную страницу, открываем этот же слот и соотносим введенные данные с имеющимися в слоте. В результате успешно выполненного теста слот будет забронирован. **Тест возможности создать отчёт о законченном слоте**. Вызываем тест авторизации. Далее убеждаемся в том, что карточка с отчётом о ближайшем слоте имеет корректные данные. Для этого сравниваем первый закончившийся слот на линейке и данные с карточки. Переходим на страницу отчётов. В переменные записываем данные, которые потом запишем в отчёт. Время квеста, количество игроков, впечатления игроков, впечатления об игроках, комментарии игроков, комментарии об игроках, внесенная сумма.

Вводим данные переменных в соответствующие элементы и сохраняем отчёт. Отчёт представлен на рисунке 7. Убеждаемся, что отчет сохранен с теми

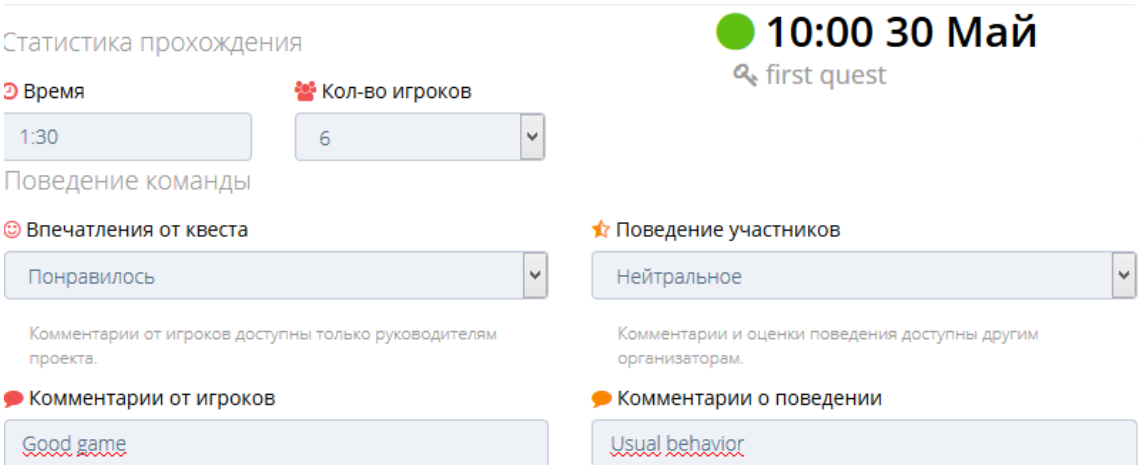

Рисунок 7 – Заполнение отчёта

данными, которые были введены. В результате успешного выполнения теста отчёт будет сформирован с введёнными данными. **Тест возможности добавить нового пользователя, используя профиль ранее зарегистрированного пользователя**. Вызываем тест авторизации, переходим на страницу прав, записываем в переменные данные, которые будет использоваться при добавлении нового пользователя и валидации данных. Открываем меню добавления нового пользователя. Заполняем его данными и сохраняем. Открываем для пользователя доступ к первому квесту. Заполнение профиля нового пользователя на рисунке 8. Убеждаемся, что профиль нового пользователя появился на странице, открываем его и сверяем имеющиеся данные с ранее введенными. Выходим из аккаунта. Авторизируемся и убеждаемся, что у пользователя есть доступ только к первому квесту и нет ко второму. **Тест возможности добавить нового пользователя, используя профиль свежезарегистрированного пользователя**. Данный тест практически полностью повторяет предыдущий. За исключением того, что успешно завершенным тест будет уже при успешной авторизации добавленного пользователя. Также этот тест не завершается удачно, т.к. сайт имеет баг в этом функционале. **Запуск и результаты автотестов** Получившийся класс позволяет как запускать автотесты поодиночке, так и все вместе. Благодаря взаимодействию Pycharm и Unittest можно сразу увидеть какие тесты были завершены удачно, а какие нет. Также можно оценить время работы тестов (см. рисунок 9). **Дальнейшее использование реализованных автотестов** Реализованные автотесты можно применять в ходе регрессионного

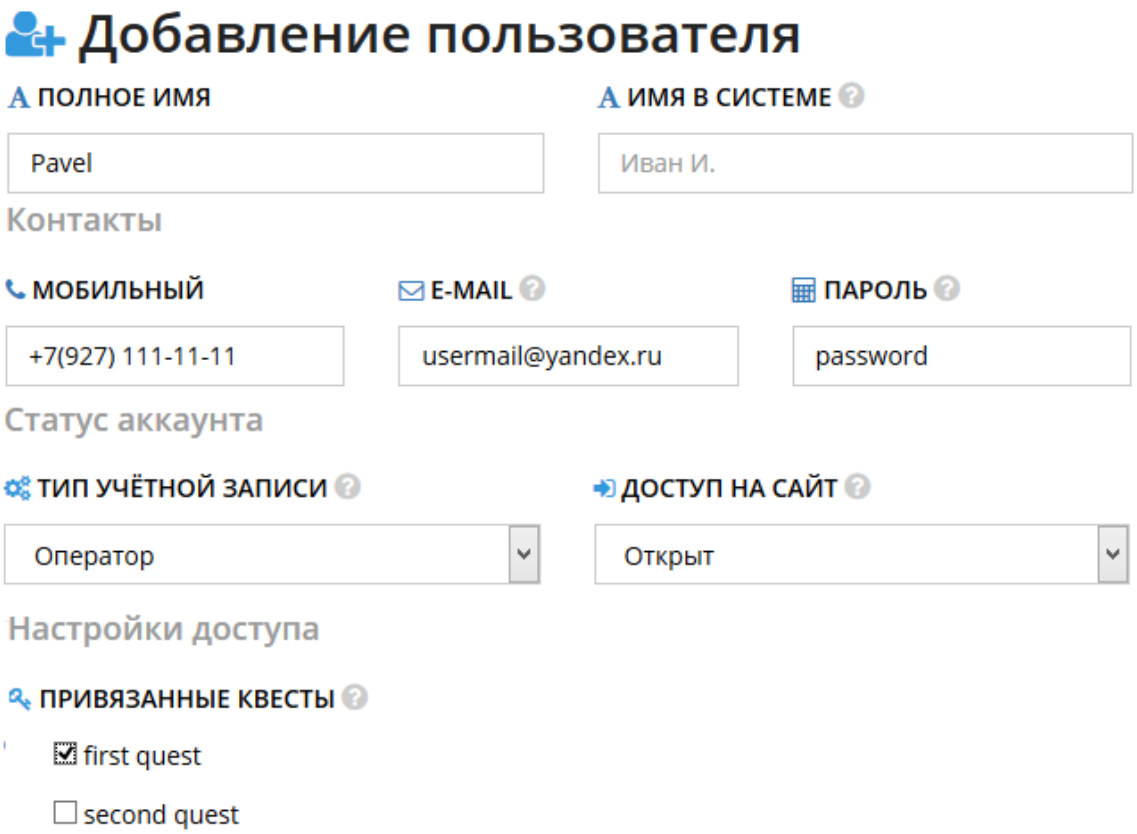

Рисунок 8 – Заполнение профиля нового пользователя

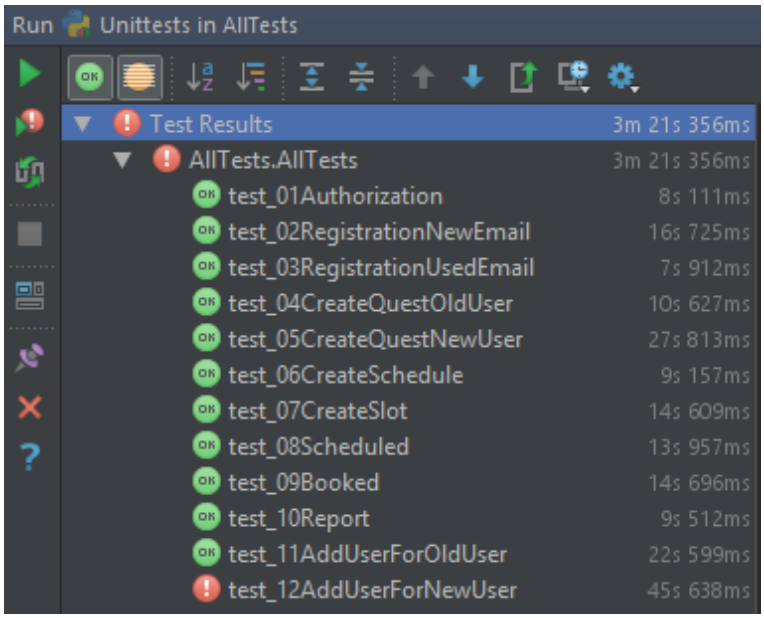

Рисунок 9 – Результаты работы автотестов в Pycharm

тестирования. Это способно существенно сэкономить время и трудозатраты. Также разработанный набор тестов можно использовать в качестве дымового тестирования. В случае изменений в GUI, затрагивающих элементы с которыми работают тесты, тесты придется редактировать.

ЗАКЛЮЧЕНИЕ

В работе были рассмотрены подходы к организации автоматизированного тестирования, достоинства и недостатки автоматизации тестирования, а также были разработаны и реализованы двенадцать автоматизированных тестов функционала сайта Quest.run. Тесты были написаны на языке Python с использованием модуля Unittest и Selenium Webdriver. С помощью данных тестов осуществляется быстрое функциональное тестирование. Также данные тесты могут использоваться после изменений проектного кода, будь то графические изменения или изменения логики проекта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 *Канер, С.* Тестирование программного обеспечения / С. Канер. — 2001.

2 Модуль Unittest [Электронный ресурс]. — URL: http://pythonworld.ru/moduli/modul-unittest.html (Дата обращения 30.05.2016). Загл. с экр. Яз. рус.

3 Selenium WebDriver [Электронный ресурс]. — URL: http://www.software-testing.ru/library/testing/functional-testing/1740 (Дата обращения 30.05.2016). Загл. с экр. Яз. рус.

4 Selenium для Python [Электронный ресурс]. — URL: http://habrahabr.ru/post/250975 (Дата обращения 30.05.2016). Загл. с экр. Яз. рус.

5 Автоматизация тестирования [Электронный ресурс]. — URL: http://www.protesting.ru/automation/functional/whytoauto.html (Дата обращения 10.06.2016). Загл. с экр. Яз. рус.

6 *Савин, Р.* Тестирование dot com / Р. Савин. — 2007.

7 Стратегия автоматизации тестирования [Электронный ресурс]. — URL: http://tproger.ru/translations/test-automation-strategy-for-agile-projects (Дата обращения 11.06.2016). Загл. с экр. Яз. рус.

8 Методы тестирования программного обеспечения [Электронный ресурс]. — URL: http://fb.ru/article/247668/metodyi-testirovaniya-programmnogo-obespecheniya/ (Дата обращения 3.06.2016). Загл. с экр. Яз. рус.

9 *Тамре, Л.* Введение в тестирование программного обеспечения / Л. Тамре. — 2003.

10 *Бейзер, Б.* Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем / Б. Бейзер. — 2004.