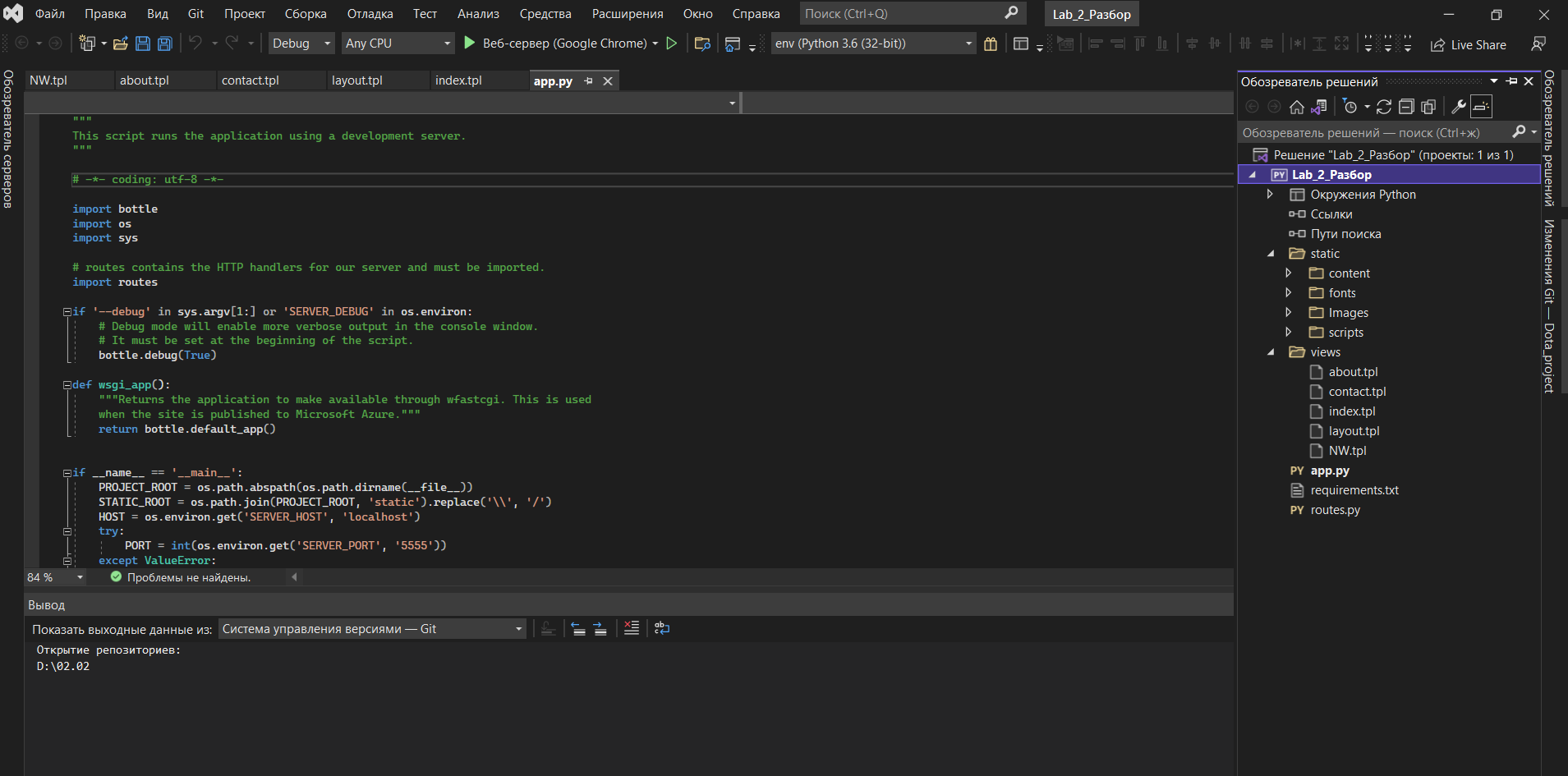
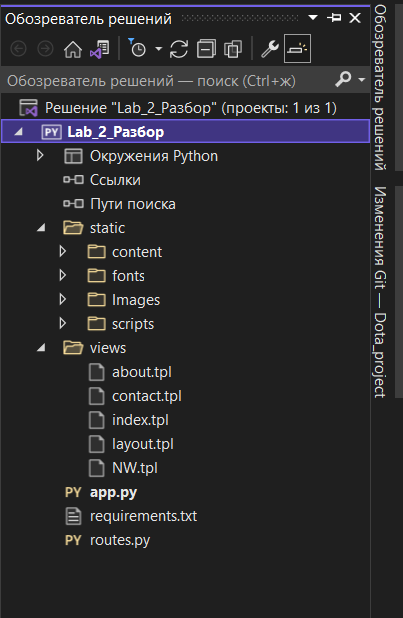
**Лабораторная работа №2-4**

**Цель**: создание модульной структуры проекта в виде диаграммы, разработка перечня артефактов (основных файлов) и протоколов, настройка Git.

1. Создание файла



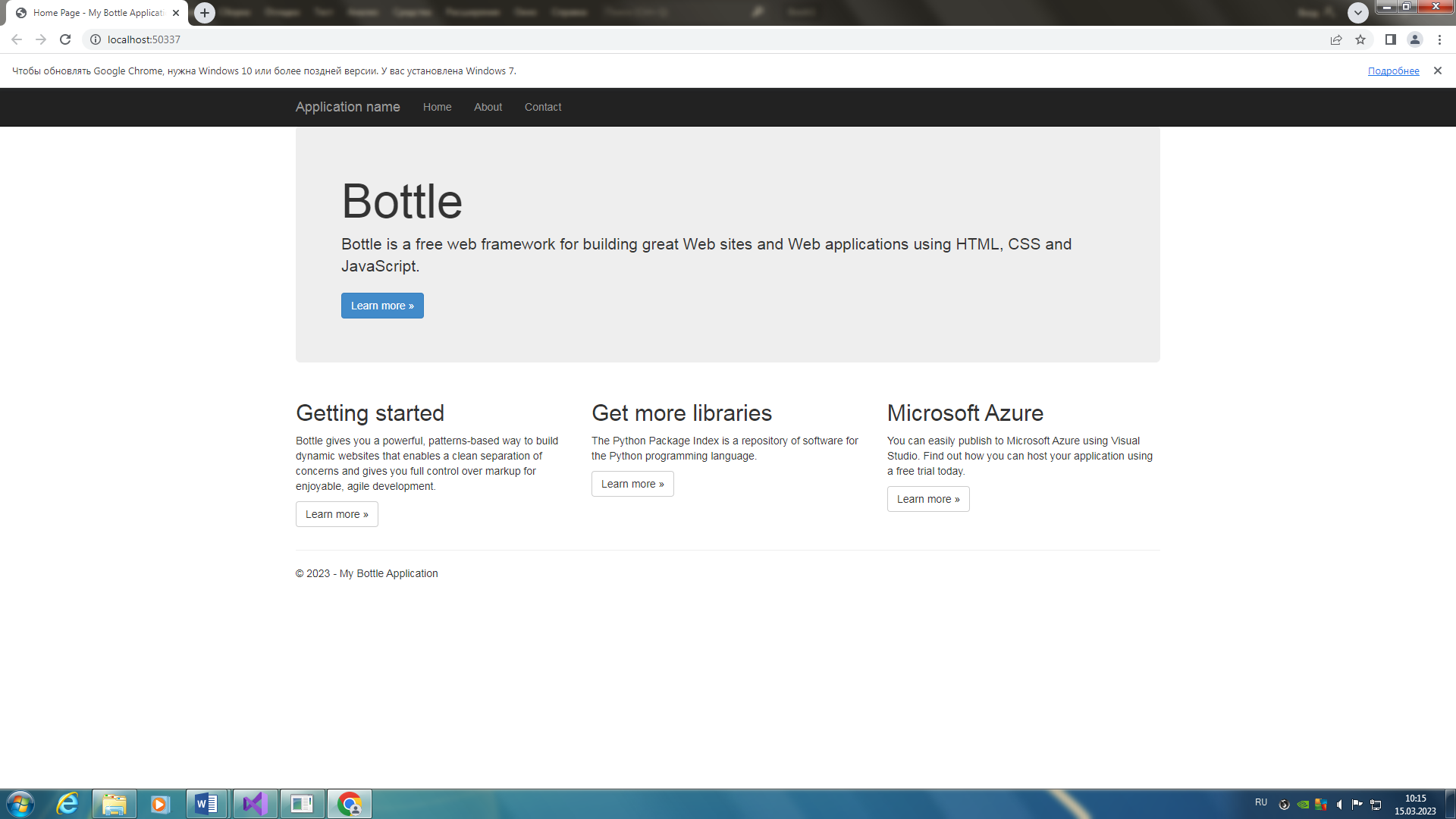
1. Обозреватель решений



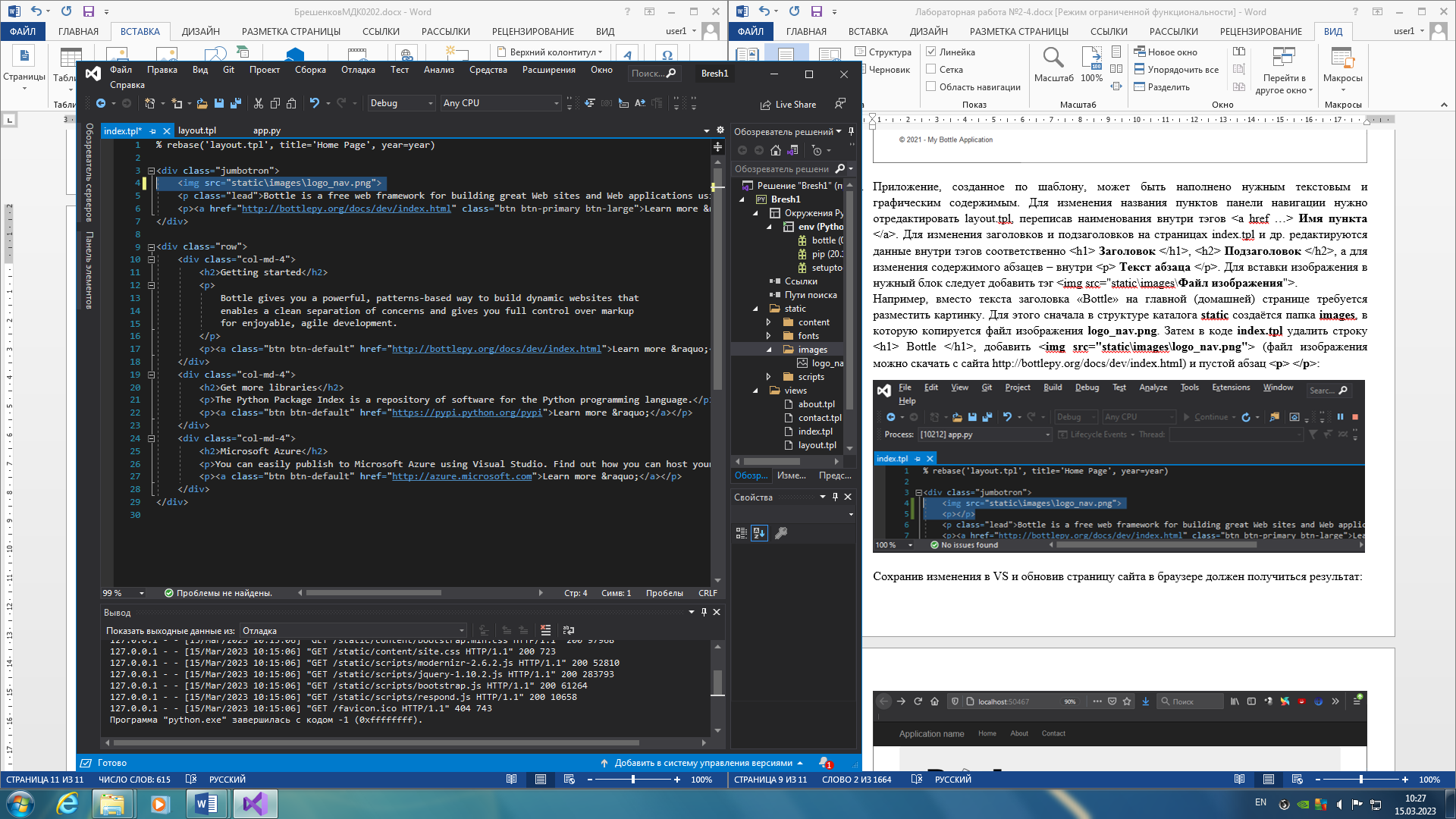
В папке **static** содержатся стилевые css-файлы для оформления внешнего вида веб-страниц, шрифтовые наборы и js-скрипты.

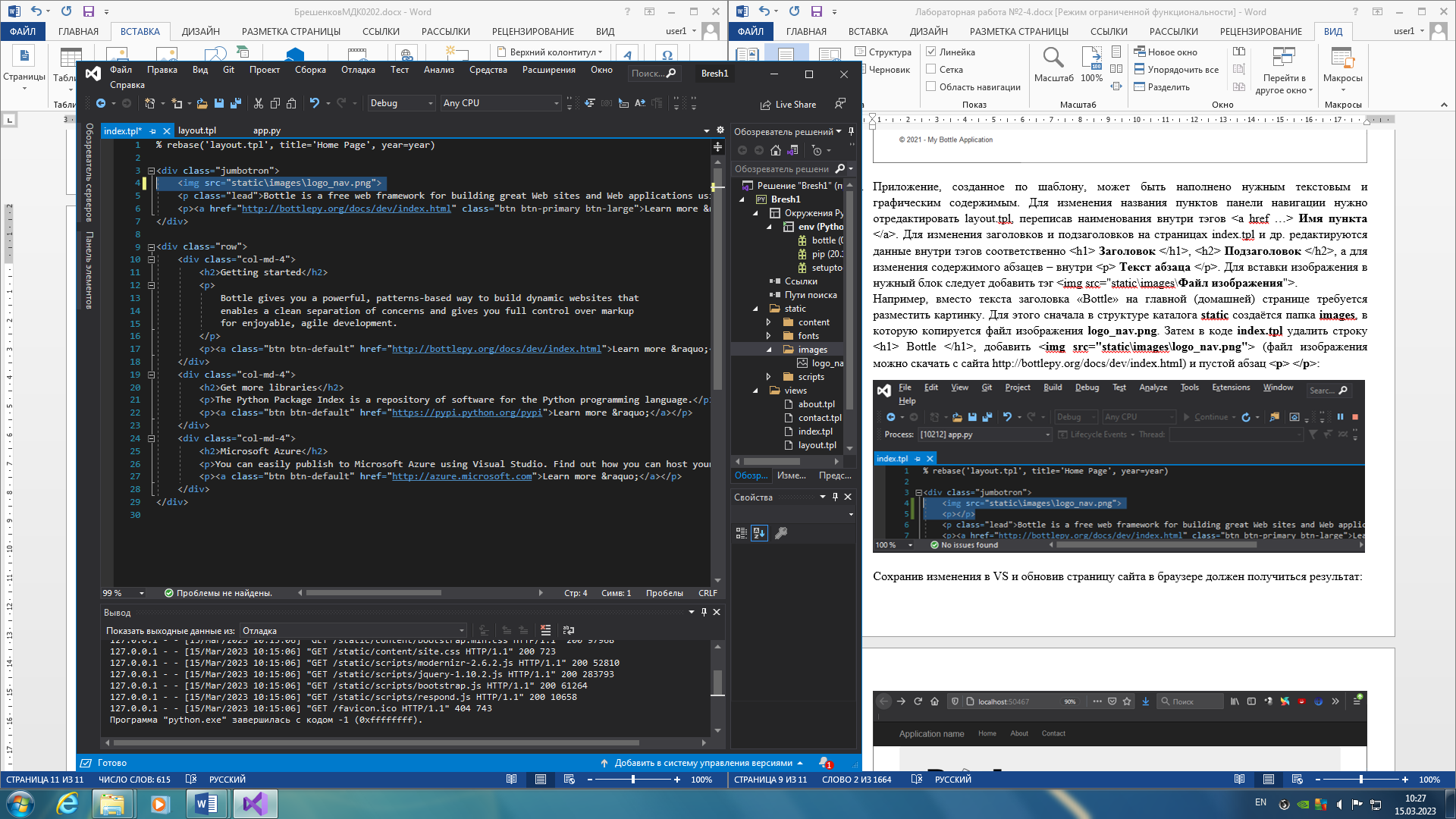
Выбрав для просмотра файлы из **папки** views, можно убедиться, что они представляют собой макеты веб-страниц: **главной** (index.tpl), **о нас** (about.tpl), **контакты** (contact.tpl), а файл layout.tpl – не что иное, как шаблон обёртки с панелью навигации, содержащей четыре гиперссылки: Application name, Home, About, Contact в «шапке» сайта и небольшим «подвалом».

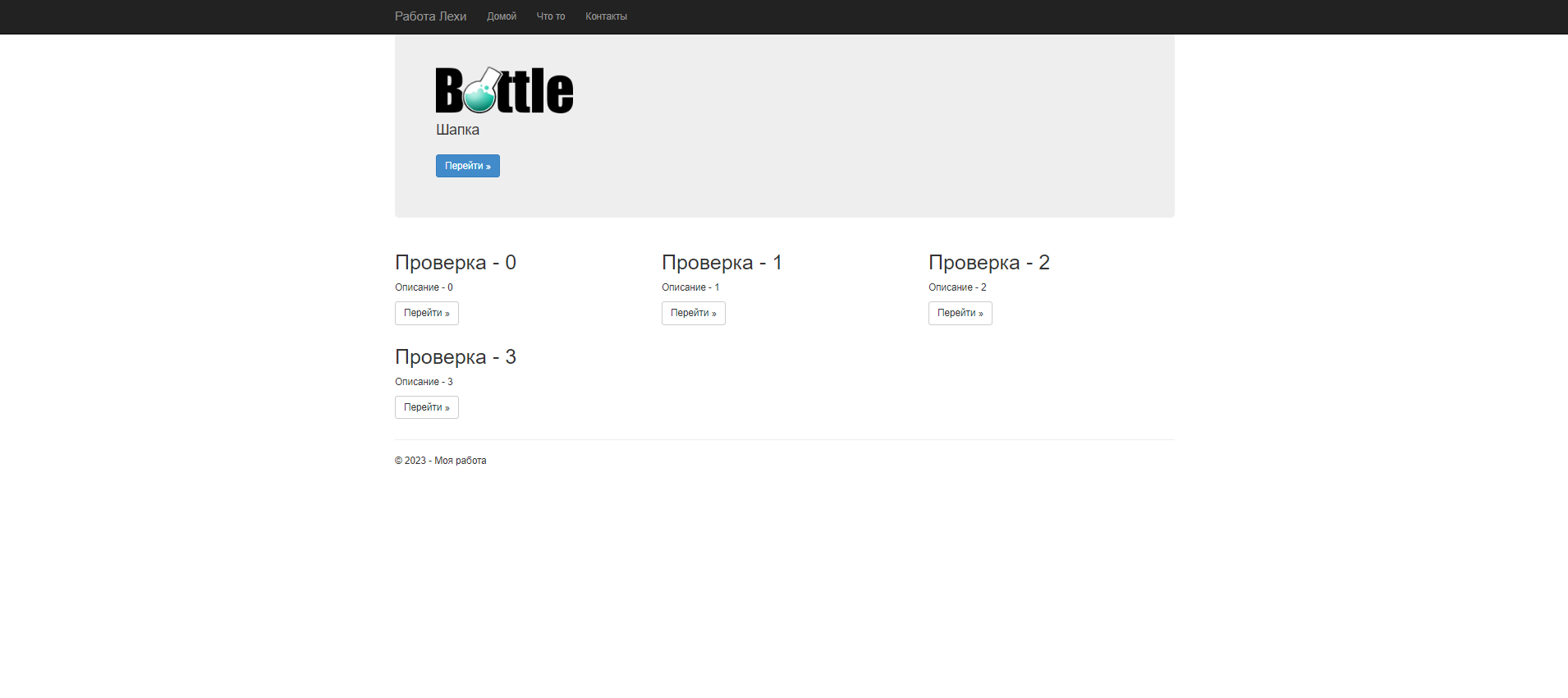
1. Выполнить запуск проекта



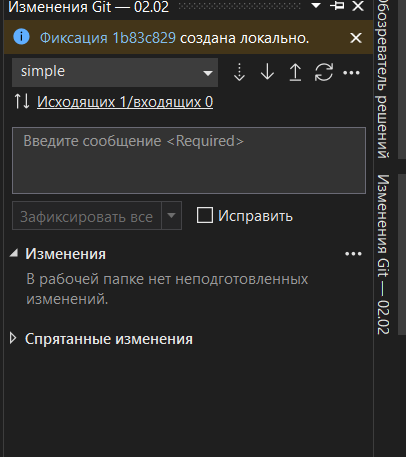
1. Добавление картинки

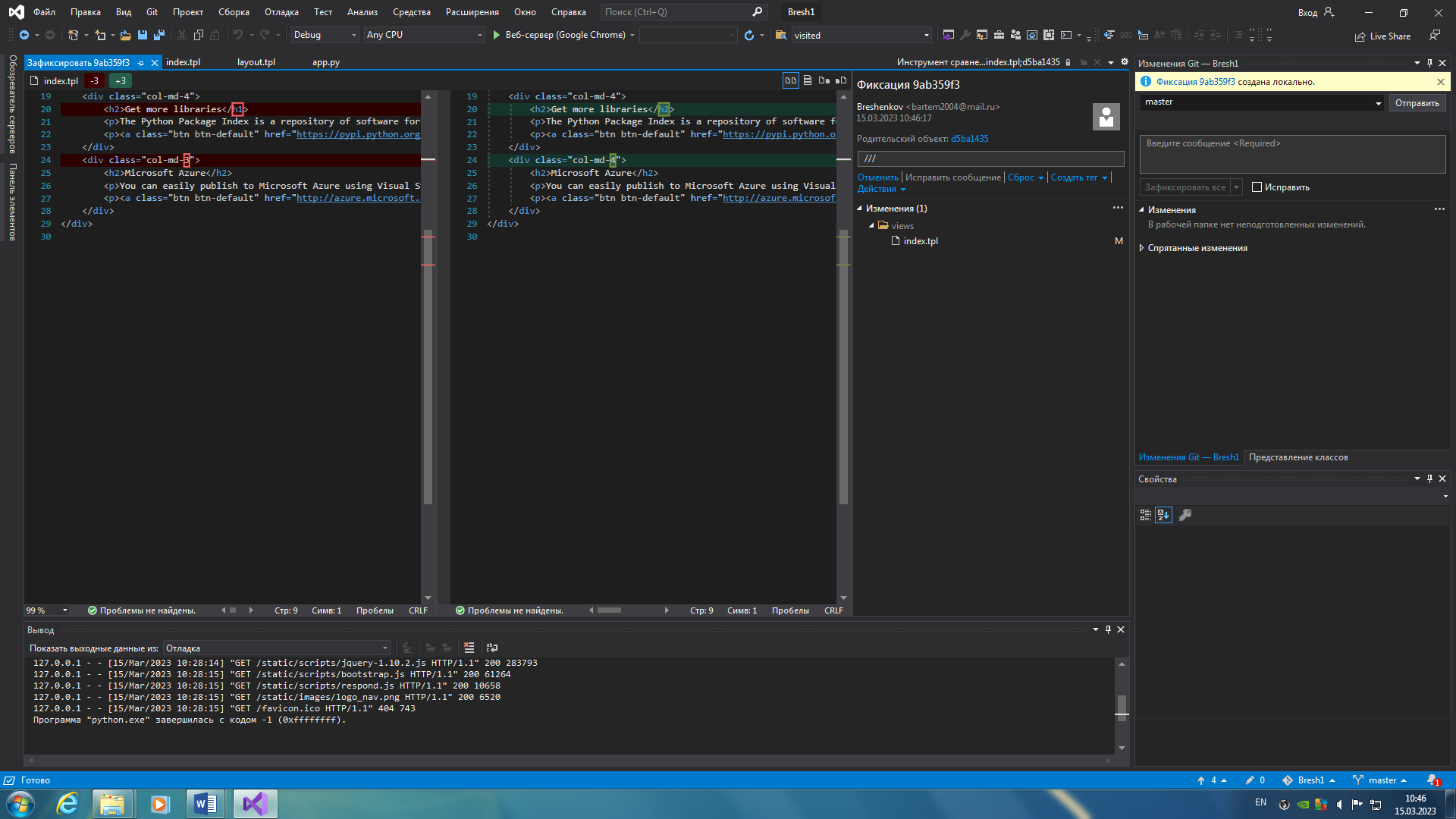


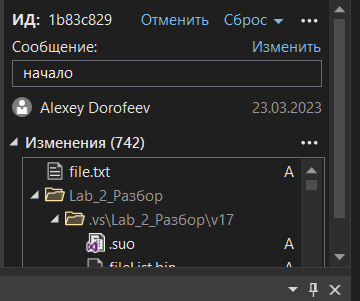




1. Git настройка



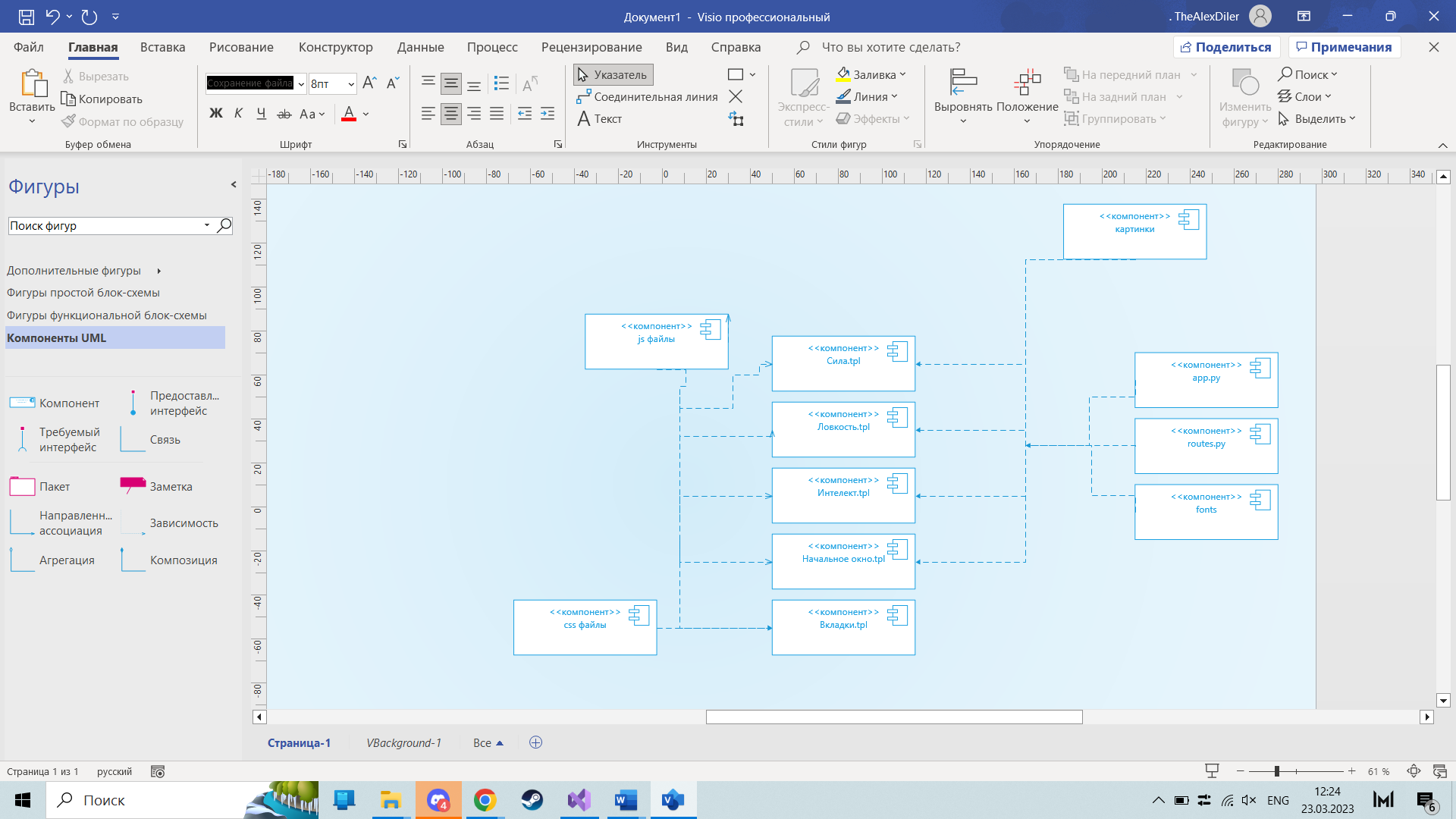




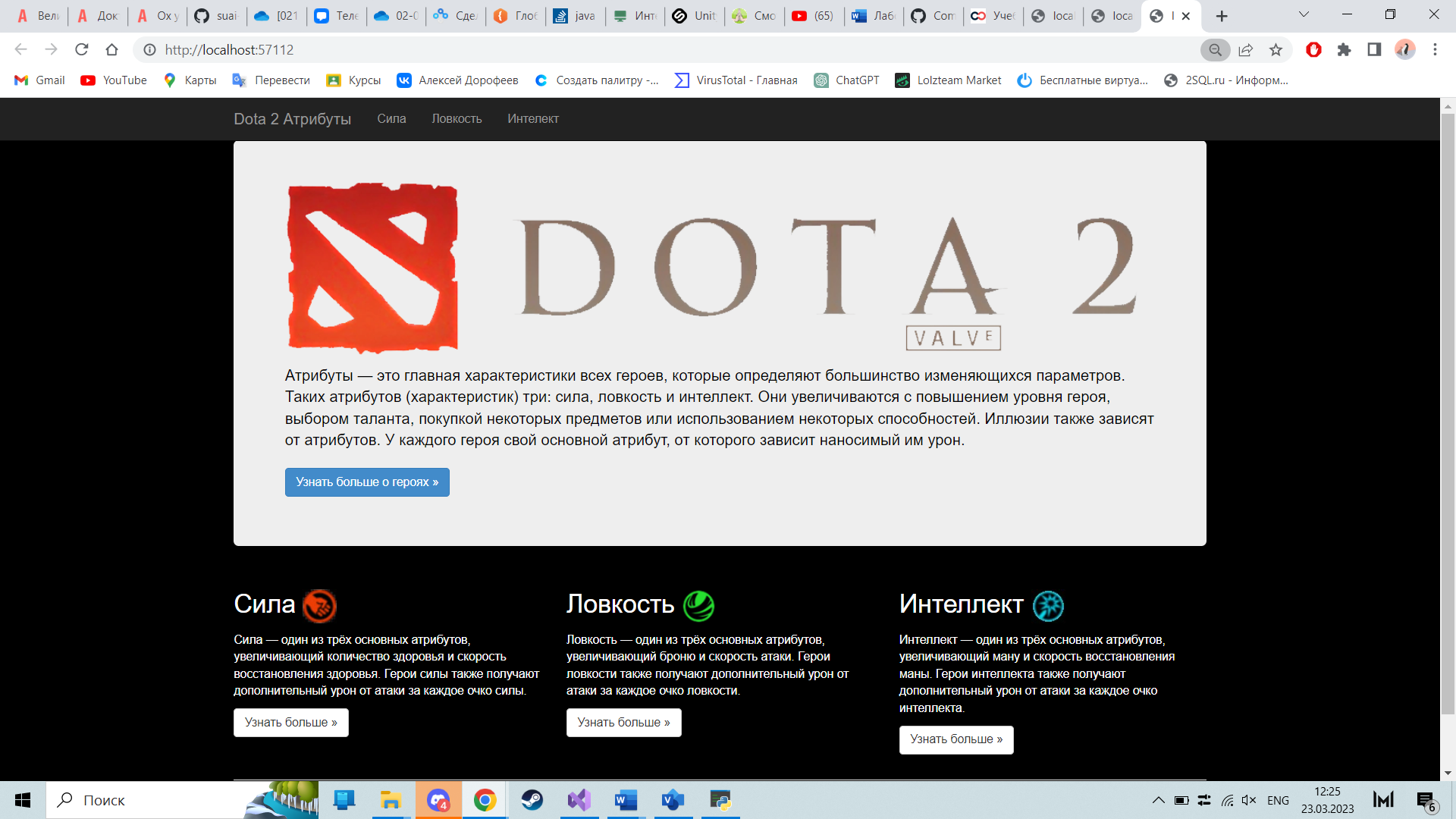
1. Требуется создать веб-приложение (тематический сайт) на основе шаблона фреймворка Bottle.

Тематика приложения: Dota 2 Атрибуты

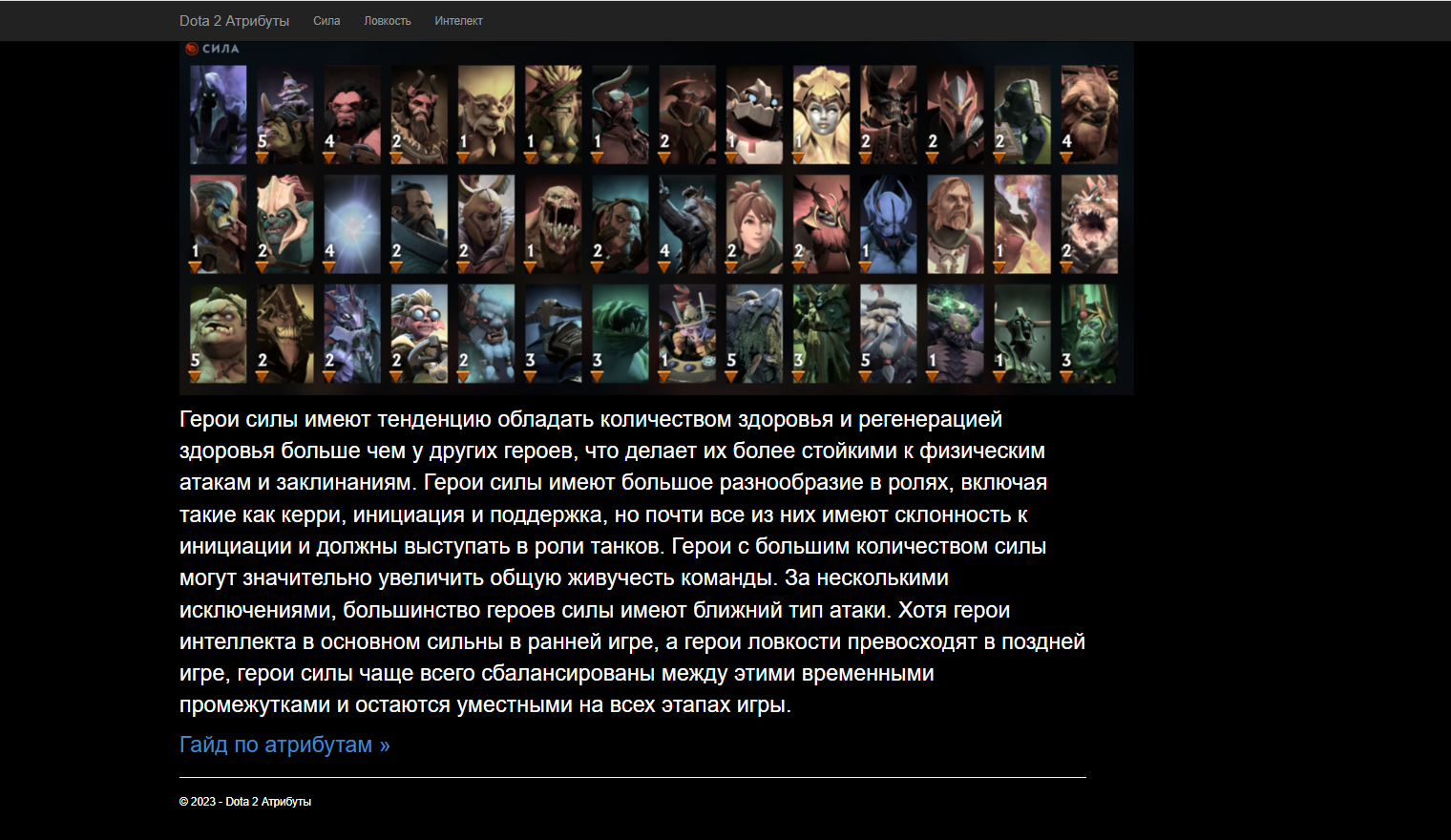
UML-диаграмма компонентов приложения:



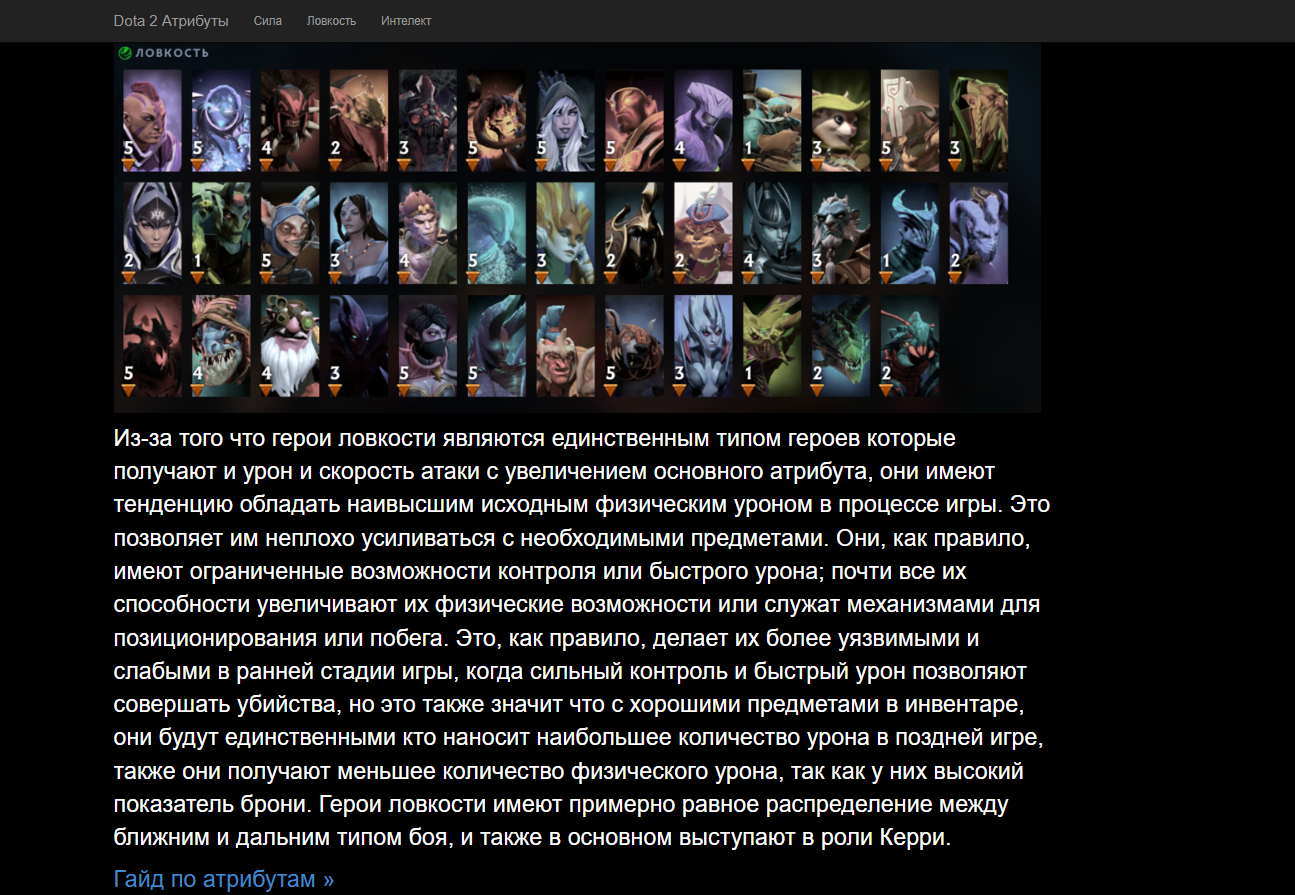
Внешний вид приложения:



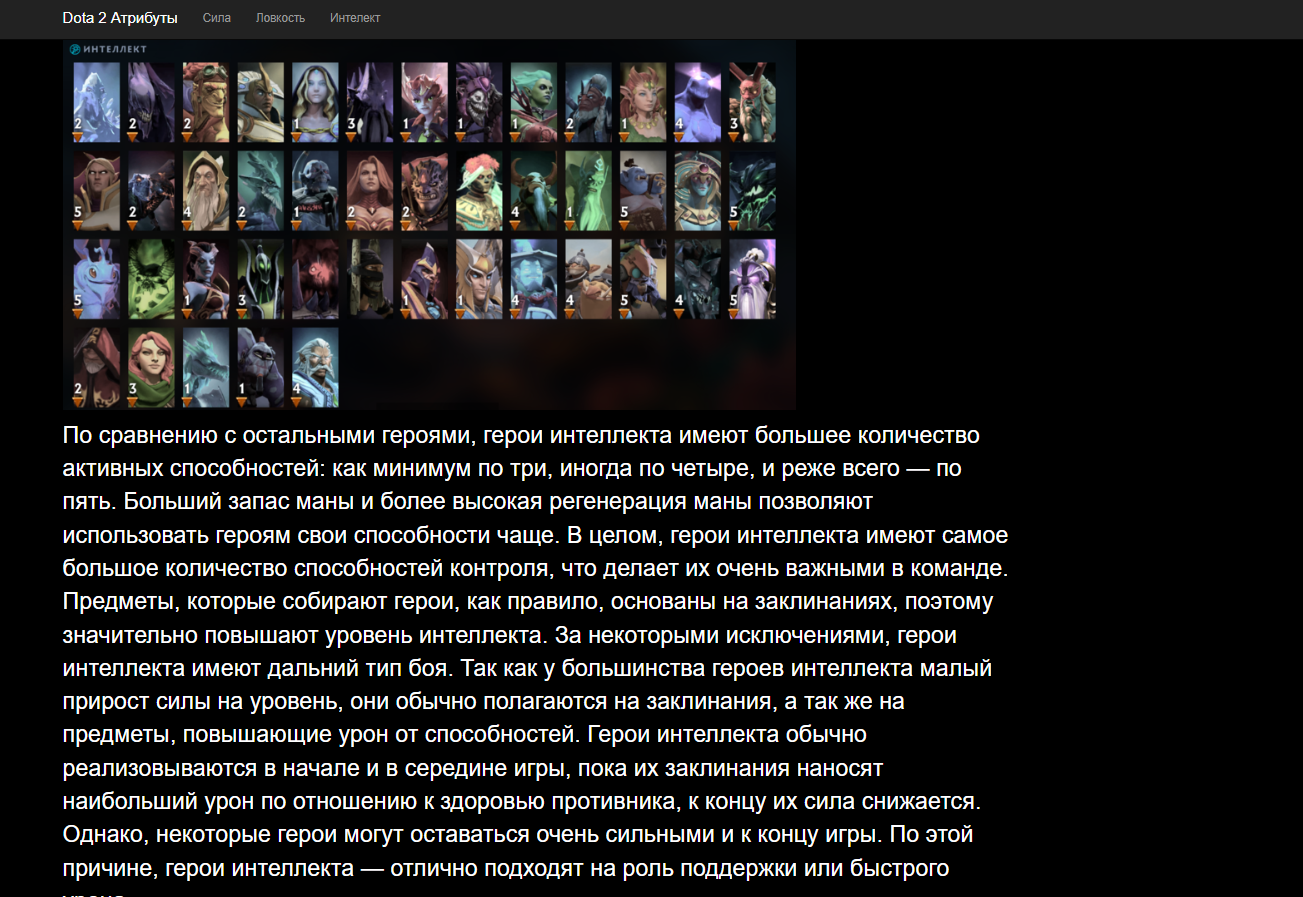
Начальная страница



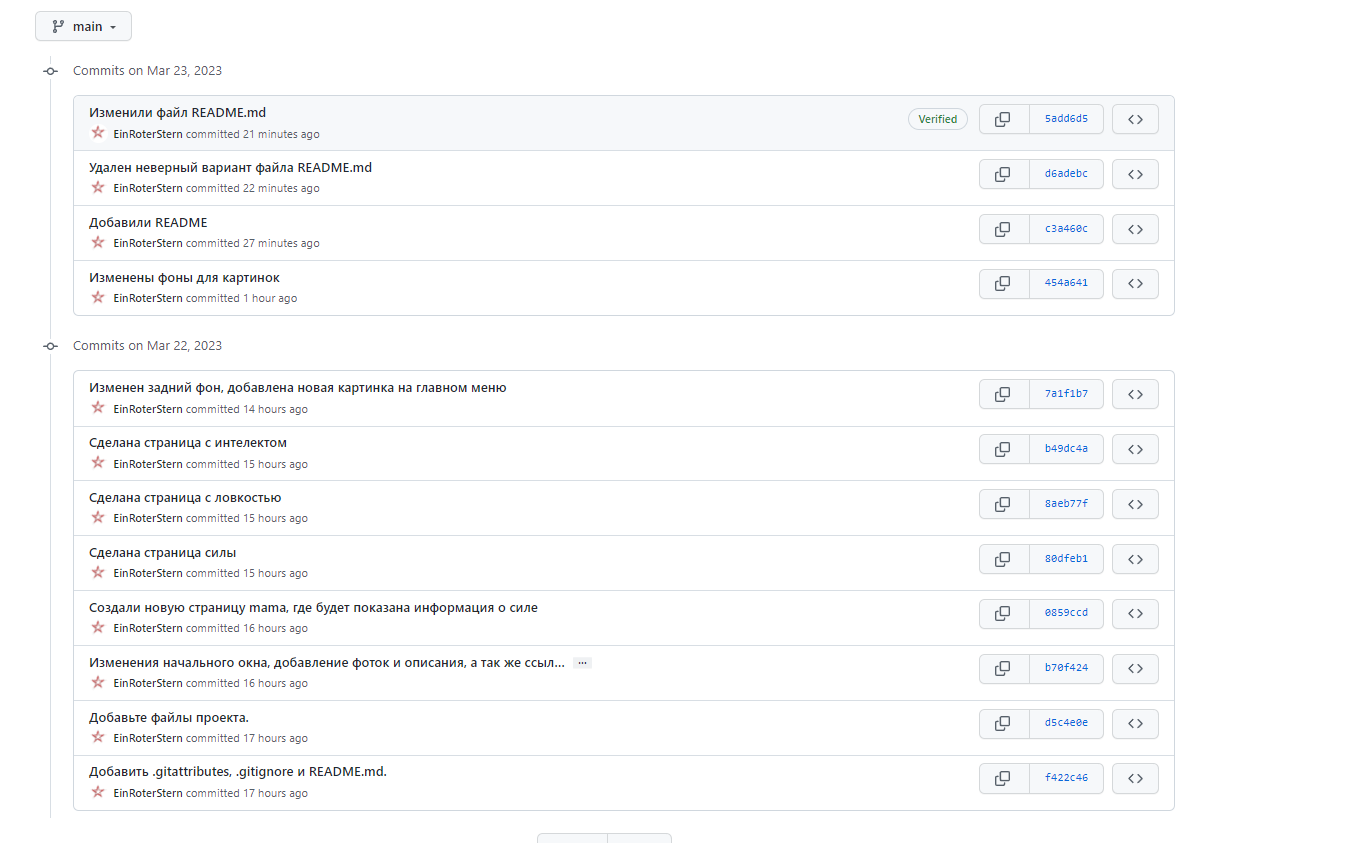
Страница Сила



Страница Ловкость

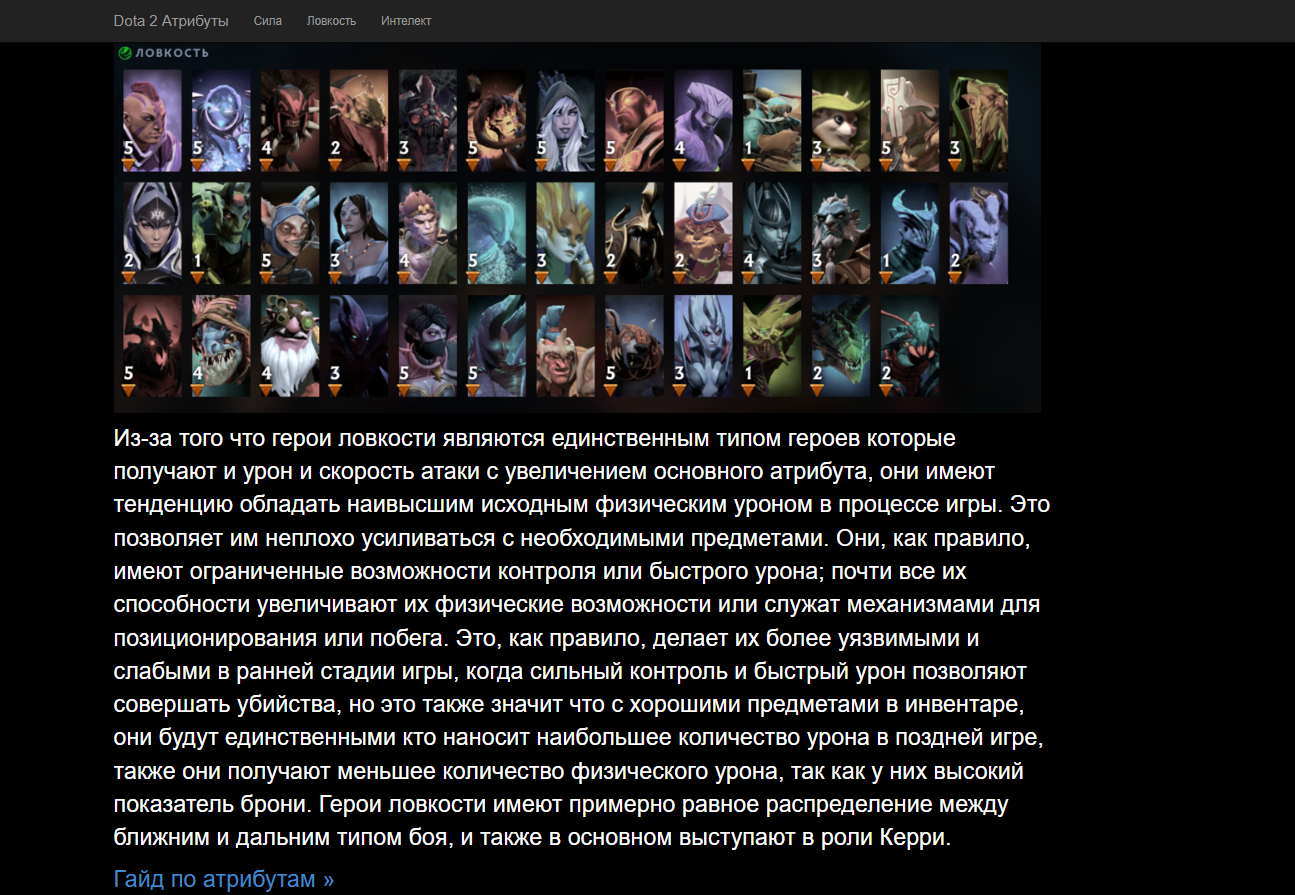


Страница Интеллект



Коммиты

Что я делал:



% rebase('layout.tpl', year=year)

<div class="Sila">

<img src="static\images\lovkost\_heroes.png">

<p></p>

<p class="siloviki">Из-за того что герои ловкости являются единственным типом героев которые получают и урон и скорость атаки с увеличением основного атрибута,

они имеют тенденцию обладать наивысшим исходным физическим уроном в процессе игры. Это позволяет им неплохо усиливаться с необходимыми предметами.

Они, как правило, имеют ограниченные возможности контроля или быстрого урона; почти все их способности увеличивают их физические возможности

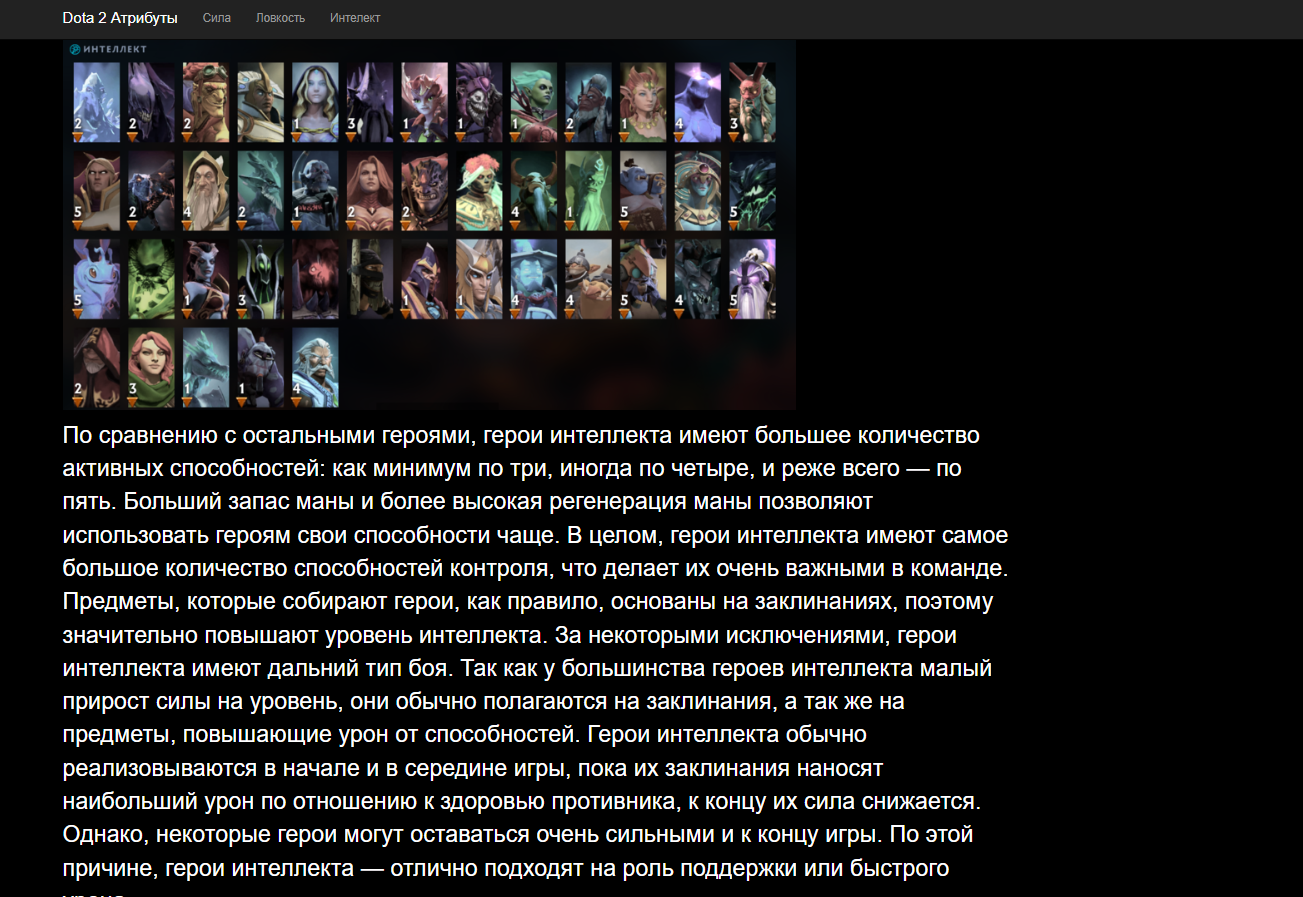
или служат механизмами для позиционирования или побега. Это, как правило, делает их более уязвимыми и слабыми в ранней стадии игры, когда сильный контроль

и быстрый урон позволяют совершать убийства, но это также значит что с хорошими предметами в инвентаре, они будут единственными кто наносит наибольшее количество урона

в поздней игре, также они получают меньшее количество физического урона, так как у них высокий показатель брони. Герои ловкости имеют примерно равное распределение

между ближним и дальним типом боя, и также в основном выступают в роли Керри.</p>

<p><a href="https://www.youtube.com/watch?v=i3YEstGdV9Y&ab\_channel=XENO" class="button\_video">Гайд по атрибутам &raquo;</a></p>

</div>

* % rebase('layout.tpl', year=year)
* <div class="Sila">
* <img src="static\images\intelect\_heroes.png">
* <p></p>
* <p class="siloviki">По сравнению с остальными героями, герои интеллекта имеют большее количество активных способностей: как минимум по три, иногда по четыре,
* и реже всего — по пять. Больший запас маны и более высокая регенерация маны позволяют использовать героям свои способности чаще. В целом, герои интеллекта
* имеют самое большое количество способностей контроля, что делает их очень важными в команде. Предметы, которые собирают герои, как правило, основаны на заклинаниях,
* поэтому значительно повышают уровень интеллекта. За некоторыми исключениями, герои интеллекта имеют дальний тип боя. Так как у большинства героев интеллекта
* малый прирост силы на уровень, они обычно полагаются на заклинания, а так же на предметы, повышающие урон от способностей. Герои интеллекта обычно реализовываются
* в начале и в середине игры, пока их заклинания наносят наибольший урон по отношению к здоровью противника, к концу их сила снижается. Однако, некоторые герои могут
* оставаться очень сильными и к концу игры. По этой причине, герои интеллекта — отлично подходят на роль поддержки или быстрого урона.</p>
* <p><a href="https://www.youtube.com/watch?v=i3YEstGdV9Y&ab\_channel=XENO" class="button\_video">Гайд по атрибутам &raquo;</a></p>
* </div>
* Контрольные вопросы:  
    
  1. Каково основное назначение UML-диаграмм и их роль в разработке приложений?  
  Основное назначение UML-диаграмм - визуализация и описание различных аспектов системы, которую нужно разработать или модифицировать. UML-диаграммы помогают увидеть общую картину и понять структуру системы, ее функциональность, взаимодействие между компонентами и объектами, а также общий процесс взаимодействия системы с внешним миром. UML-диаграммы являются важным инструментом для коммуникации между разработчиками и заинтересованными сторонами проекта.  
    
  2. Можно ли считать артефактами веб-страницы сайта? Обоснуйте ответ.  
  Да, веб-страницы сайта можно считать артефактами. Веб-страницы являются результатом работы веб-разработчика, и они представляют собой конечный продукт разработки. Кроме того, веб-страницы могут быть использованы в качестве артефактов для оценки качества разработки, например, при тестировании.  
    
  3. Предпочтительнее ли протокол HTTPS протоколу HTTP? Почему (да / нет)?  
  Да, предпочтительнее использовать протокол HTTPS вместо HTTP. Это связано с тем, что HTTPS обеспечивает более высокий уровень защиты данных, потому что он использует шифрование для защиты данных, передаваемых между сервером и клиентом. HTTPS также обеспечивает подтверждение подлинности сервера, что предотвращает атаки типа "man-in-the-middle".  
    
  4. Что значит код ответа сервера «200»? Какие ещё группы и коды ответов вы знаете (привести по 1-2 примера из каждой группы)?  
  Код ответа сервера "200" означает успешный ответ на запрос клиента. Еще группы и коды ответов, которые могут быть получены в ответ на запрос клиента, включают:  
  Коды ответов 300-399 - перенаправление, например, "301 Moved Permanently" или "302 Found".  
  Коды ответов 400-499 - ошибка на стороне клиента, например, "404 Not Found" или "403 Forbidden".  
  Коды ответов 500-599 - ошибка на стороне сервера, например, "500 Internal Server Error" или "503 Service Unavailable".  
    
  5. Какую систему защиты и сертификации данных использует Git по умолчанию?  
  Git использует систему защиты и сертификации данных под названием GPG (GNU Privacy Guard) по умолчанию. GPG используется для подписи и проверки целостности коммитов в репозитории, что помогает предотвратить внесение нежелательных изменений в репозиторий.  
    
  6. Что значит «клонировать» репозиторий в терминах Git?  
  "Клонирование" репозитория в терминах Git означает создание локальной копии удаленного репозитория Git на вашем компьютере. Клонирование позволяет получить полную копию удаленного репозитория на локальном устройстве, включая все истории коммитов, ветки и теги.

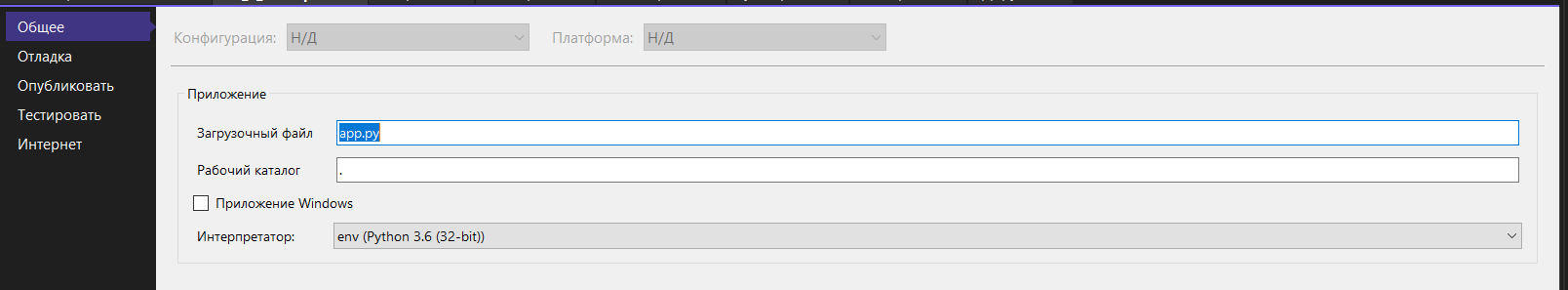
**Лабораторная работа №5, 6**

**Разработка и интеграция модулей проекта. Отладка отдельных модулей ПП.**

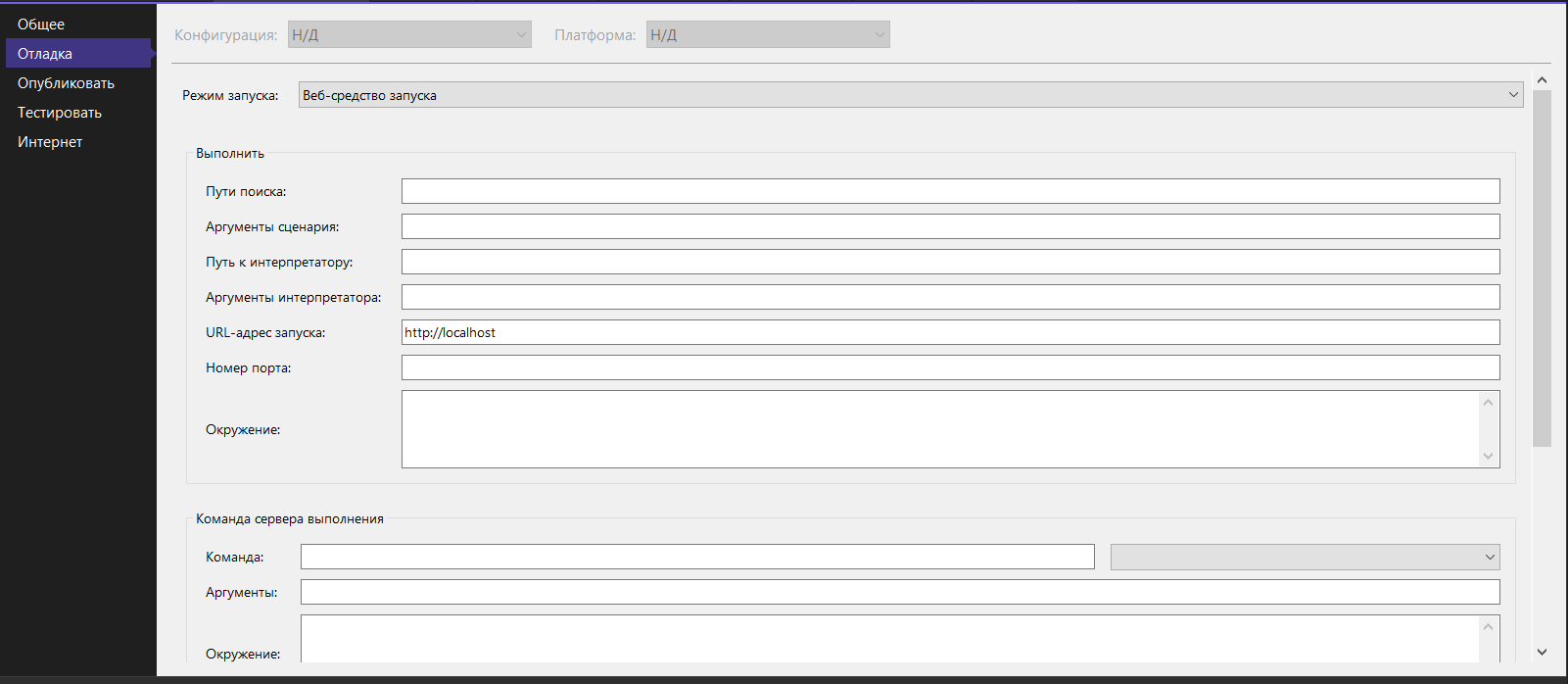
**Организация обработки исключений.**

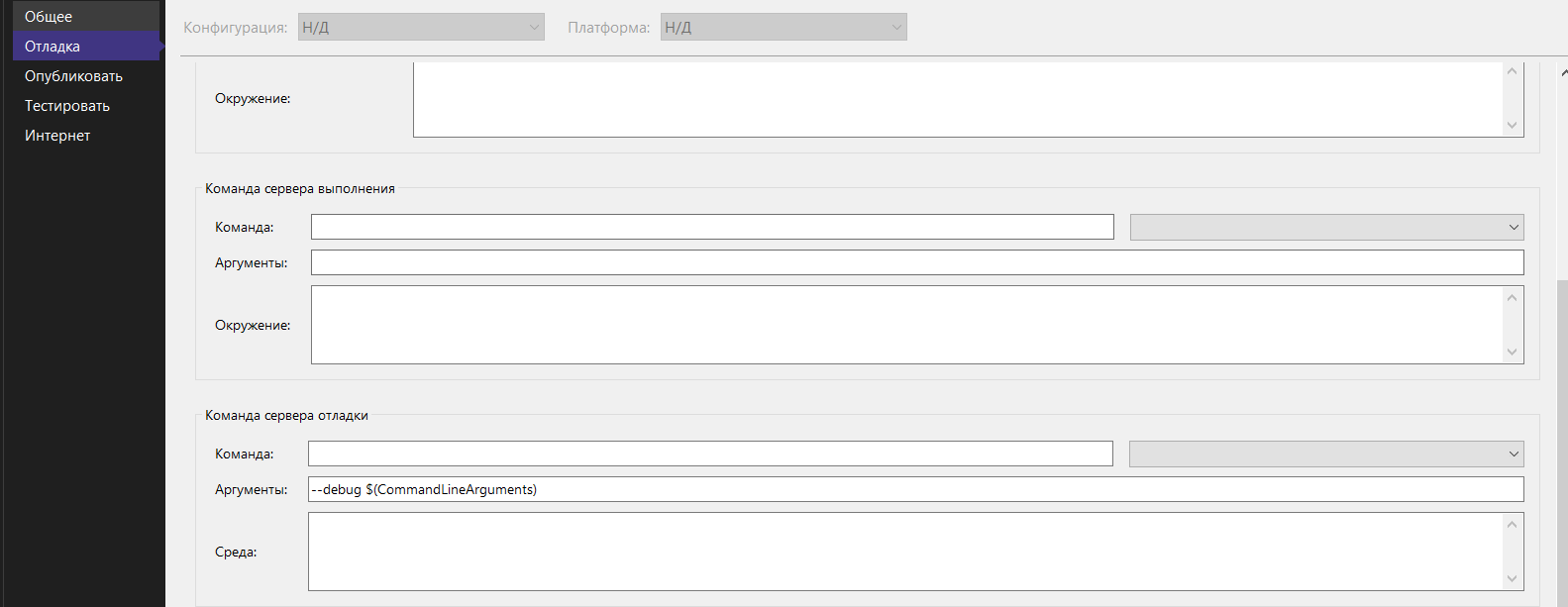
Цель: создание дополнительного модуля проекта, его интеграция в систему, выполнение отладки и обработки исключительных ситуаций.

1. В открывшемся окне свойств отладки с помощью всплывающих подсказок изучить описания параметров и те их значения, которые заданы по умолчанию. Результаты внести в отчёт.



общее



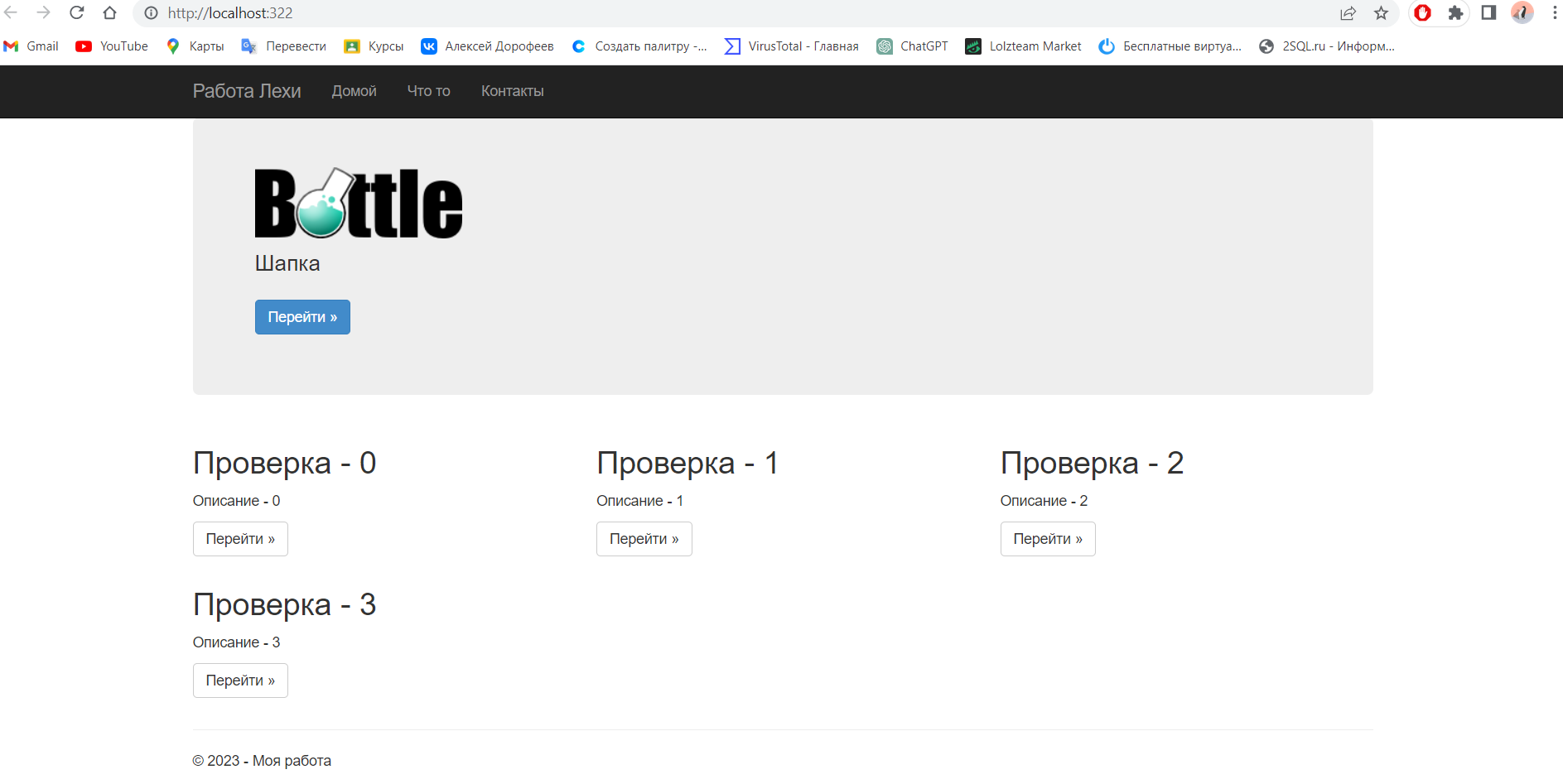


Откладка

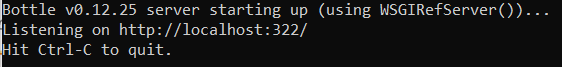
1. Установить фиксированный номер порта, например, 322:



Установка порта



Порт на сайте





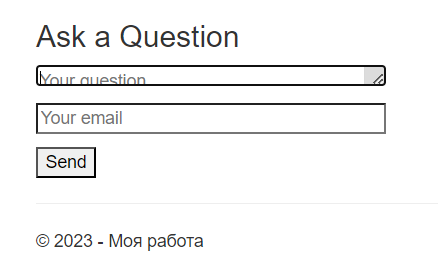
Удалили порт



Порт на сайте после удаления

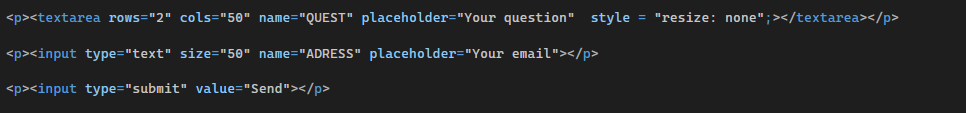


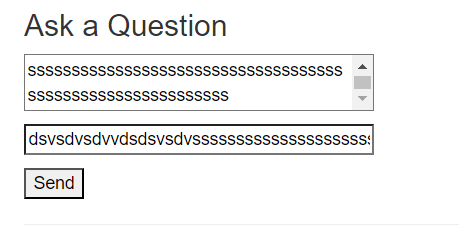
1. Добавить на главную страницу сайта простейшую форму обратной связи, состоящую из текстовой области, текстового поля и кнопки.



Выяснить, что означают атрибуты HTML-тэгов, сделать размер textarea неизменяемым.

Style = “resize: none”;



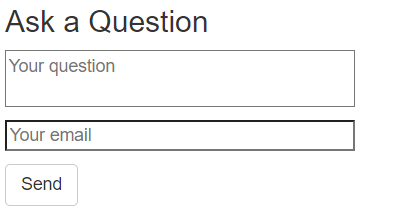


После фиксации

Т.к. кнопка отличается по стилю, то следует в соответствующий её тэг добавить

class="btn btn-default". Обновить страницу, проверить результат.





После изменения кнопки

1. Теперь необходимо добавить в папку проекта файл-обработчик для формы. Для этого сначала остановить отладку, затем добавить новый объект, выполнив Проект > Добавить … (Add New Item…). В открывшемся списке выбрать Пустой файл Python, ввести имя файла myform.py

Код в файле myfrom.py:

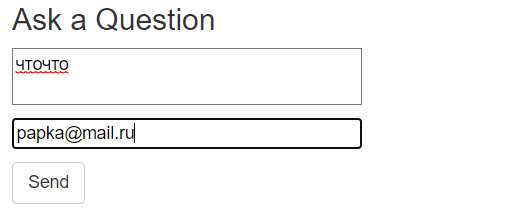
from bottle import post, request

@post('/home', method='post')

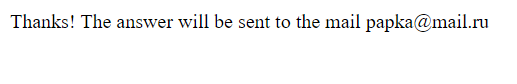
def my\_form():

mail = request.forms.get('ADRESS')

return "Thanks! The answer will be sent to the mail %s" % mail



Заполнение полей



Результат после нажатия на кнопку

* 1. Дописать в файл обработчик:
  2. Паттерн для адреса электронной почты (проверку на соответствие формату).

pattern = r"^[-\w\.][+@([-\w]+\.)+[-\w]{2,4}$](mailto:+@([-\w%5d+\.)+%5b-\w%5d%7b2,4%7d$)" – регулярное выражение

#проверяем правильность почты

if not re.match(pattern, mail):

#print(f"Invalid email: {mail}") # логируем ошибку в консоль

response.status = 400

return template('index.tpl', message='dont corect email', year=datetime.now().year)

* 1. Обращение по имени в результирующем сообщении, например, «Thanks, Nik! …», предварительно дописав необходимый элемент вёрстки «USERNAME» в шаблоне страницы. При тестировании вводить своё имя, набранное на латинице.
  2. Дату обращения пользователя в сокращённом формате согласно текущей системной дате в конце результирующего сообщения, например, «… Access Date: 2023-02-22».



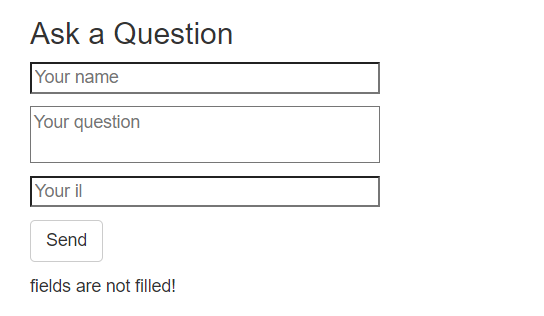
* 1. Проверку заполненности полей формы (если хотя бы одно поле не заполнено выводить соответствующее сообщение).

#проверяем заполненость полей

if not username or not mail or not question:

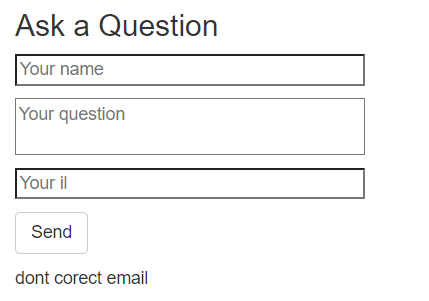
return template('index.tpl', message='fields are not filled!', year=datetime.now().year)

Не заполнено/ы поле/я:

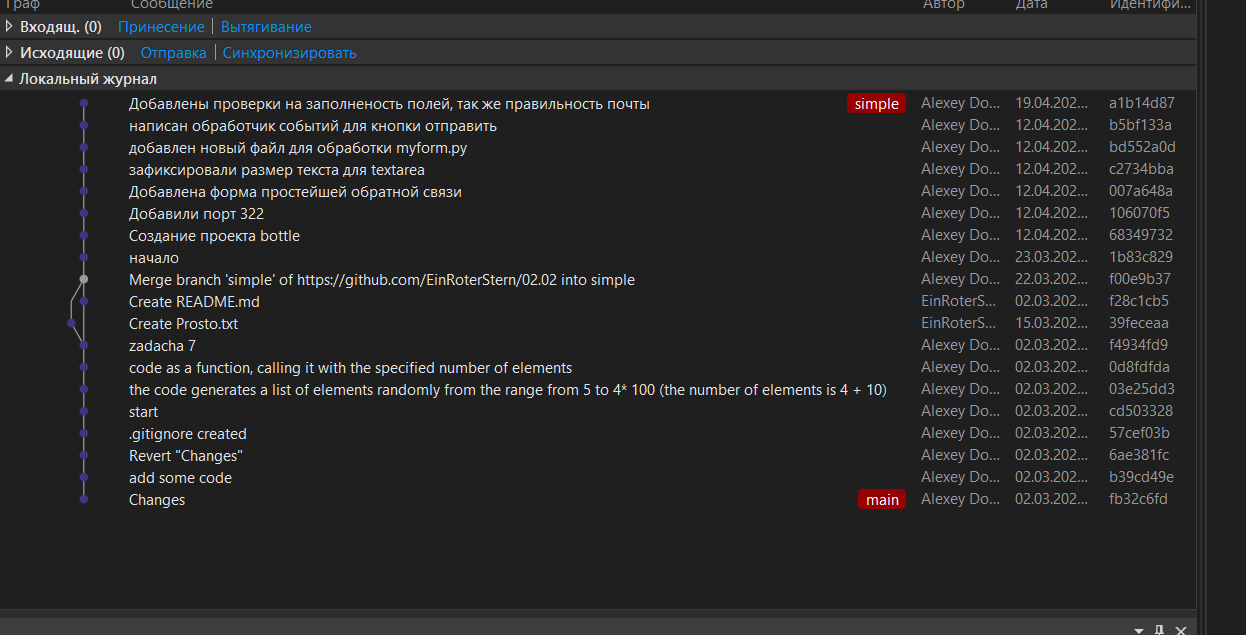


Почта не верном формате

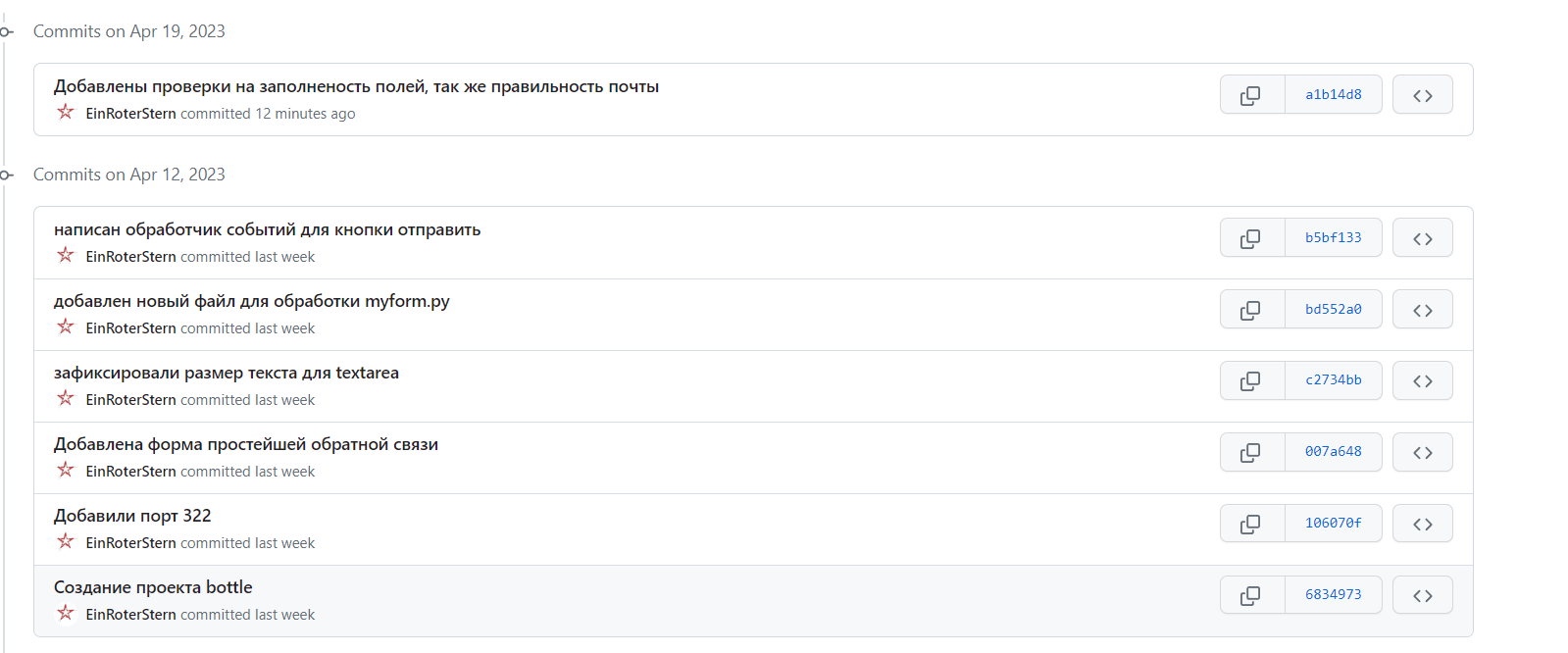




История коммитов:



Коммиты на GitHub:



Контрольные вопросы:

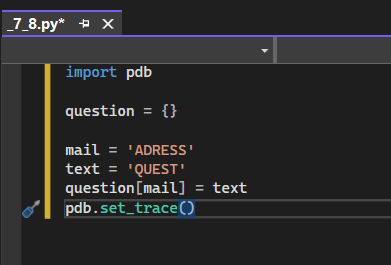
1. Группа "Отладка" вкладки "Веб-средство запуска" включает следующие параметры:  
- "Отладочный режим": включает или выключает режим отладки. Если включен, то при возникновении ошибок на странице будет выводиться отладочная информация.  
- "Автоматическая перезагрузка приложения": включает или выключает автоматическую перезагрузку приложения при изменении файлов в папке с приложением.  
- "Открыть консоль разработчика при возникновении ошибок": открывает консоль разработчика в браузере при возникновении ошибок.  
- "Журнал отладки": выводит сообщения журнала отладки в консоль разработчика.  
\ \ \  
2. Самые известные искл ситуации:  
- TypeError: возникает при использовании неверных типов данных в операциях, например, попытка сложения строки и числа.  
  
- ValueError: возникает, когда функция получает аргументы правильного типа, но некорректных значений.  
  
- FileNotFoundError: возникает, когда файл не может быть найден в системе.  
  
- ZeroDivisionError: возникает при делении на ноль.  
  
3. Регулярное выражение: \+7\s\(\d{3}\)\s\d{3}-\d{2}-\d{2} – должно начинаться с +7, s – пробельный символ, (d{3}) – 3 цифры подряд, также открытие и закрытие скобок, d{3}-\d{2}-\d{2} – три цифры подряд дефис две цифры дефис две цифры

**Лабораторная работа №7, 8**

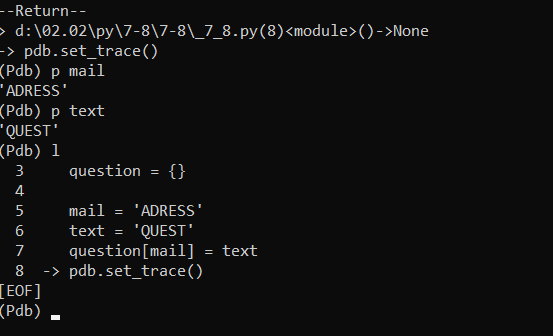
**Тема**: Применение отладочных классов в проекте, его отладка и инспекция кода модулей.

**Цель**: использование технологий и инструментов отладки и тестирования, встроенных в Visual Studio для выполнения тестирования разного типа с целью обнаружения ошибок и их дальнейшее исправление, ведение документации и анализ результатов тестирования.

1. Создание консоли и ввод кода:

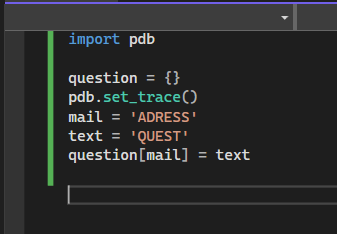


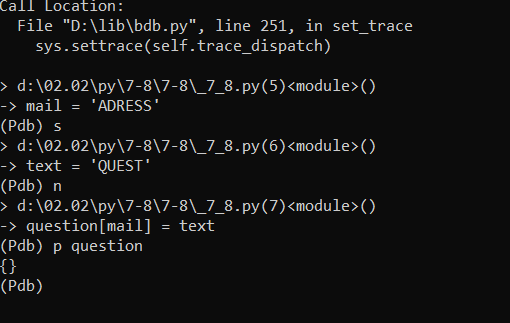
Запуск откладки



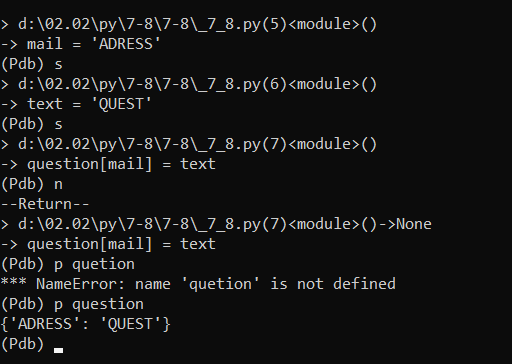
1. Перемещение инструкции pdb.set\_trace() на 3 строки выше и выполнение кода

построчно:

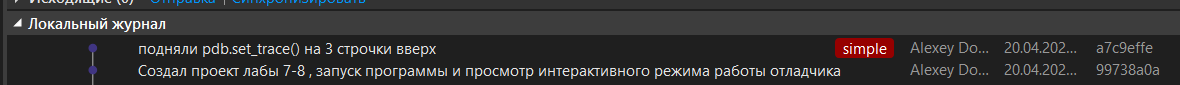


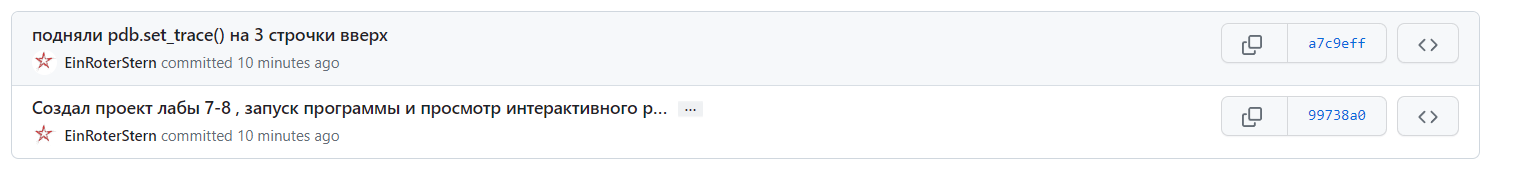


1. Исправление ситуации с пустым словарем:

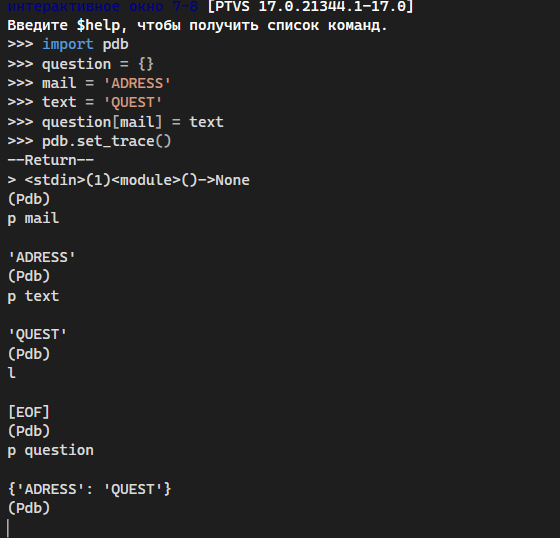


Коммиты:

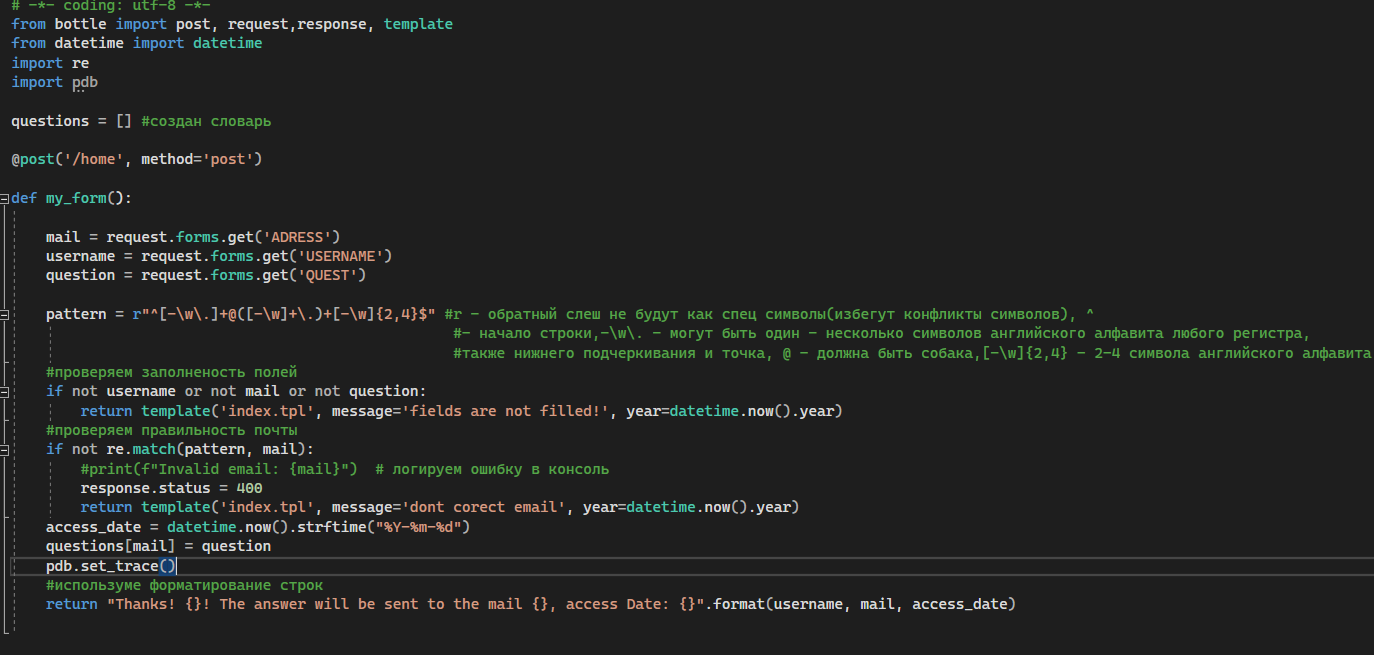




1. Выполните пошаговое выполнение кода с помощью интерактивного окна окружения Python, результаты внесите в отчёт. Закройте проект консольного приложения.



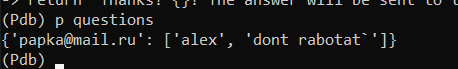
1. Откройте проект, в котором в предыдущей ЛР на главную страницу сайта была добавлена форма. Допишите код файла с функцией обработки формы таким образом, чтобы данные, вводимые пользователем, записывались в словарь (email – ключ, question – значение). Для демонстрации результата используйте отладочную печать библиотеки pdb:



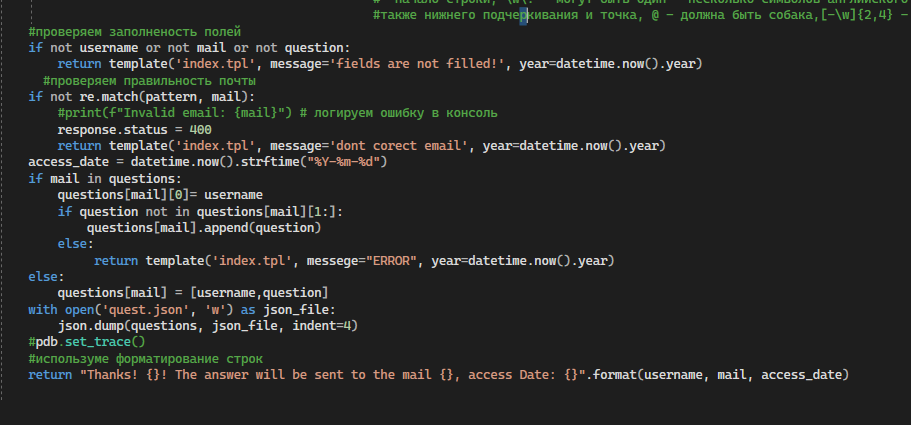


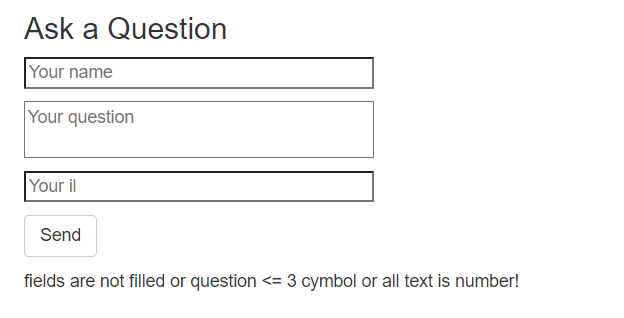
1. Измените словарь таким образом, чтобы в качестве значения по ключу mail считывался список [«USERNAME», «QUESTION»]. Затем используйте консольный отладчик.

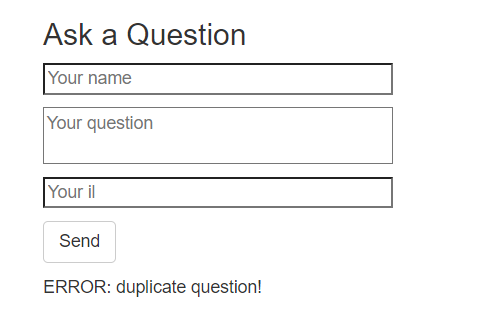


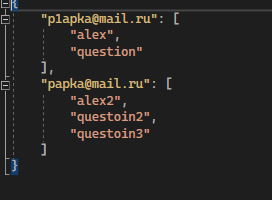


1. Выполните запись данных формы в файл JSON-объекта. Протестируйте несколько раз, вводя новые данные (файл должен дополняться). При изменении файла проверяйте как почту пользователя (у одного и того же пользователя вопросы записываются в список, соответствующий его почтовому адресу), так и содержание вопроса (дубликаты не записываются).  Кроме того, в поле вопроса при заполнении число символов – более 3-х, а текст не должен состоять из одних цифр. Допишите соответствующие проверки. Все результаты внесите в отчёт.

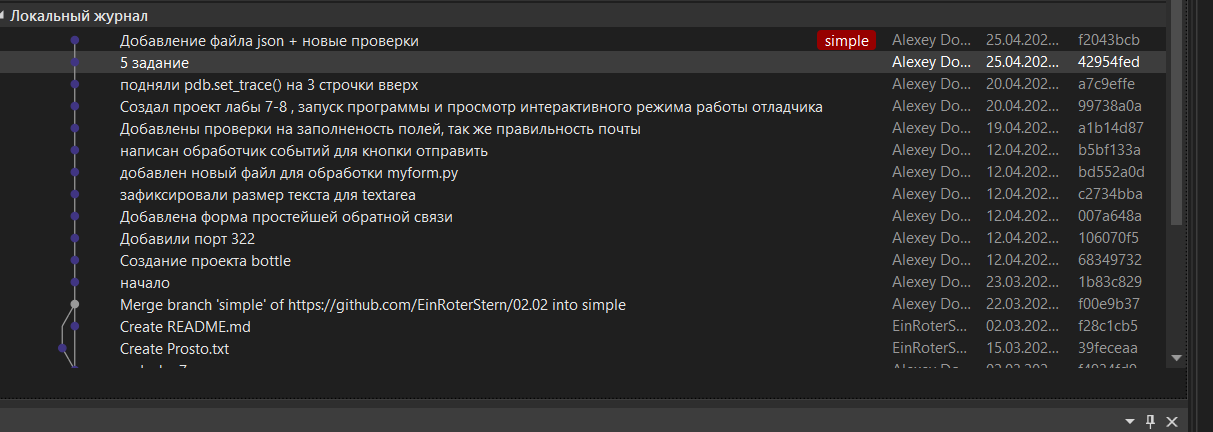








Коммиты:



Контрольные вопросы:

1. Какие методы отладки и тестирования вы знаете?

* Тестирование модульное: проверка отдельных модулей программы на соответствие ожидаемому поведению.
* Тестирование интеграционное: проверка взаимодействия между различными модулями или компонентами программы.
* Тестирование системное: проверка всей системы в целом на соответствие требованиям и ожидаемому поведению.
* Тестирование функциональное: проверка функциональности программы на соответствие спецификациям и ожидаемому поведению.
* Тестирование нагрузочное: проверка реакции системы на высокие нагрузки и определение ее производительности.
* Тестирование безопасности: проверка системы на уязвимости и возможные атаки.

1. Какие возможности предоставляют команды библиотеки pdb? Назовите

команды.

n (next): выполнить следующую строку кода.

c (continue): продолжить выполнение программы до следующей точки останова или до завершения.

q (quit): прервать выполнение программы и выйти из отладчика.

l (list): вывести исходный код вокруг текущей строки выполнения.

p (print): вывести значение переменной.

s (step): выполнить текущую строку и остановиться в подпрограмме, если она есть.

h (help): вывести справку по командам отладчика pdb.

1. Расшифруйте аббревиатуру JSON. Дайте определения этому формату.

JSON "JavaScript Object Notation" - это текстовый формат обмена данными,

основанный на подмножестве синтаксиса языка JavaScript. Он широко

используется для передачи структурированных данных между клиентом и

сервером в веб-приложениях.

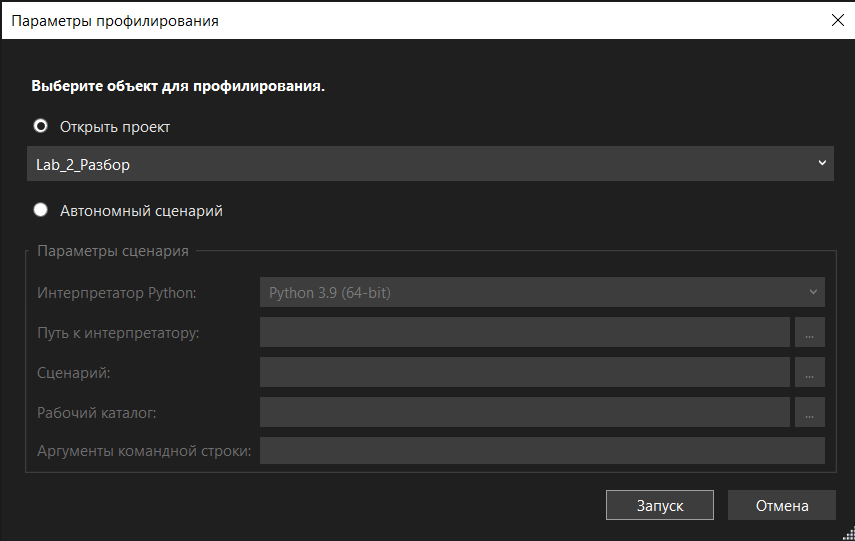
1. Поддерживается ли JSON протоколом обмена SOAP?

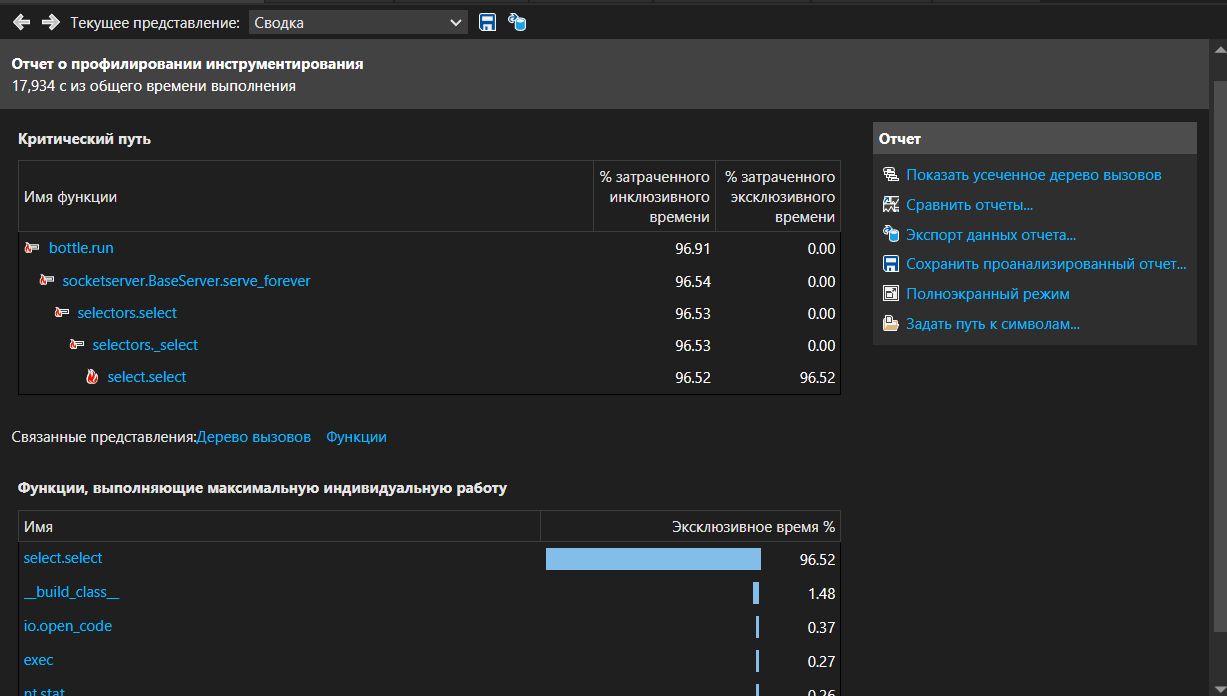
Нет, JSON не поддерживает протокол обмена SOAP. SOAP (Simple Object Access Protocol) - это протокол обмена структурированными сообщениями, основанный на XML (eXtensible Markup Language). В то время как JSON также является форматом обмена данными, он не использует XML и не взаимодействует с протоколом SOAP.

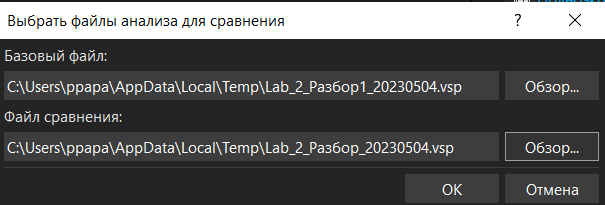
**Лабораторная работа №9-12**

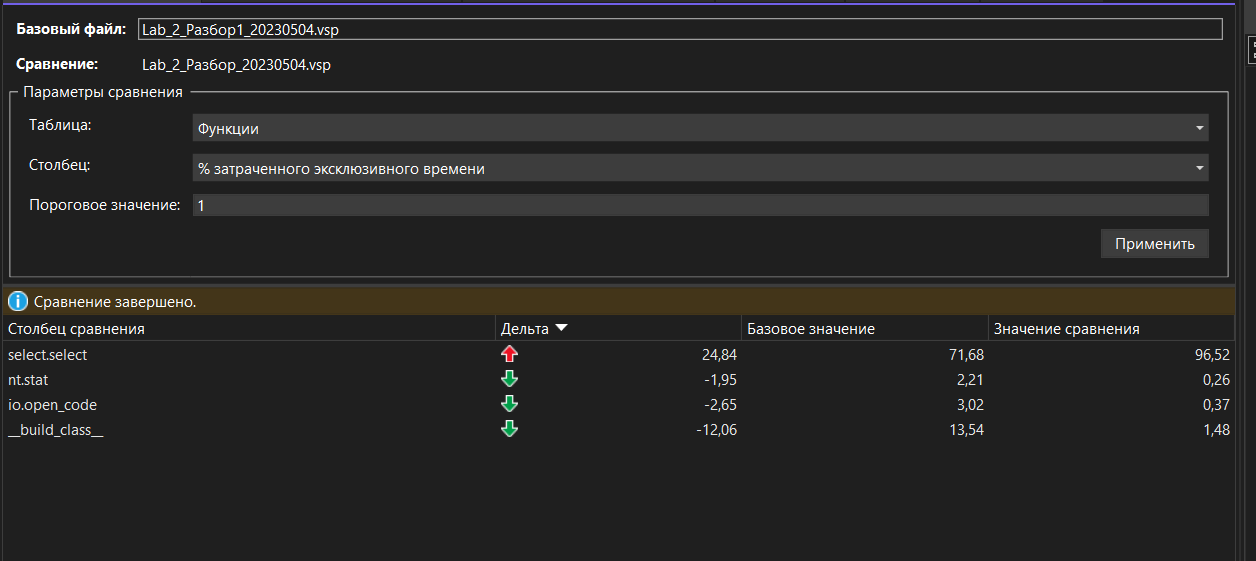
**Тема**: Профилирование кода средствами инструментальной среды разработки. Разработка тестовых модулей проекта. Выполнение функционального тестирования и тестирования интеграции. Документирование результатов. Тестирование интерфейса пользователя.

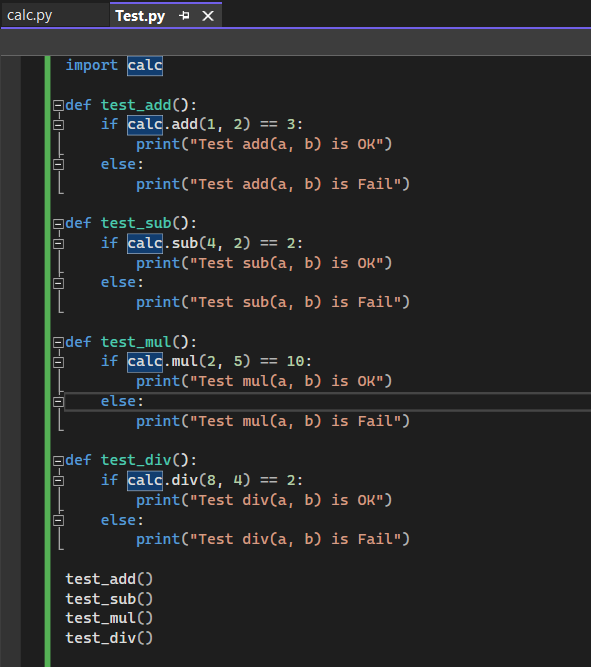
**Цель**: использование технологий и инструментов отладки и тестирования, встроенных в Visual Studio для выполнения тестирования разного типа с целью обнаружения ошибок и их дальнейшее исправление, ведение документации и анализ результатов тестирования.

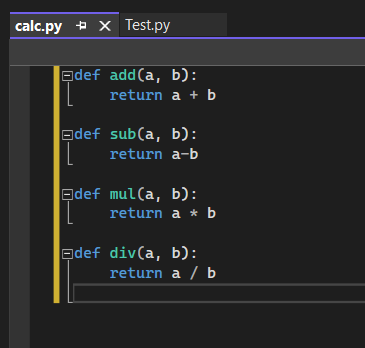


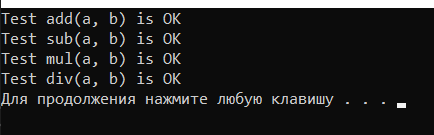


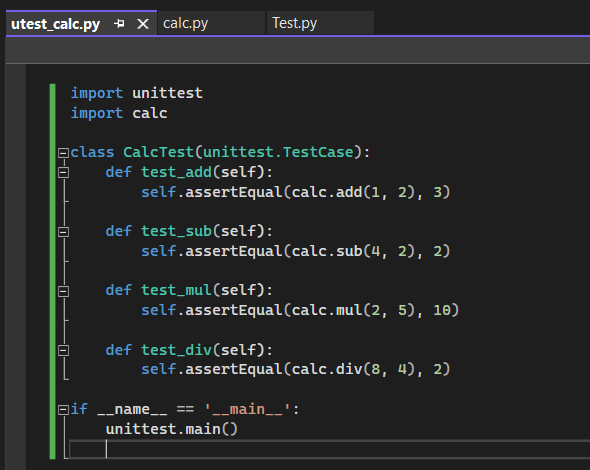


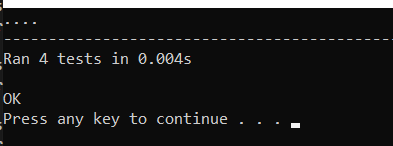


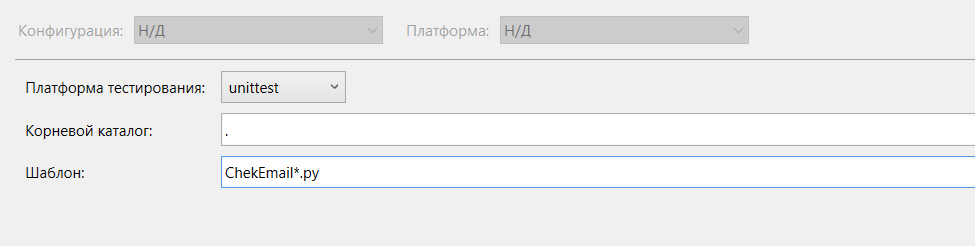












assertFalse

import unittest

from mail import check\_email as mail

class CheckEmail(unittest.TestCase):

def test\_incorect\_mail(self):

mails = ["","1","g3g@","@gmail", "g@gmail.", ".d@gmail..c", "@@gmail", "gm@@gmail.com",

"user@example", "user@.example.com", "user.@example.com", "user..example.com",

"user@-example.com", "user@example-.com", "user@111.222.333.444", "user@.255.255.255.255",

"user@example..com", "user@example.com.", "user@ex\_ample.com" ]

for email in mails:

self.assertFalse(mail(email))

def test\_corect\_mail(self):

mails\_cor = [

"john.doe@example.com",

"jane\_doe@example.co.uk",

"james1234@example.net",

"bob-smith@example.org",

"alice+test@example.com",

"test@example.com",

"user123@example.edu",

"example123@example.com",

"user.name@example.com",

"support@example.com"

]

for email in mails\_cor:

self.assertTrue(mail(email))



