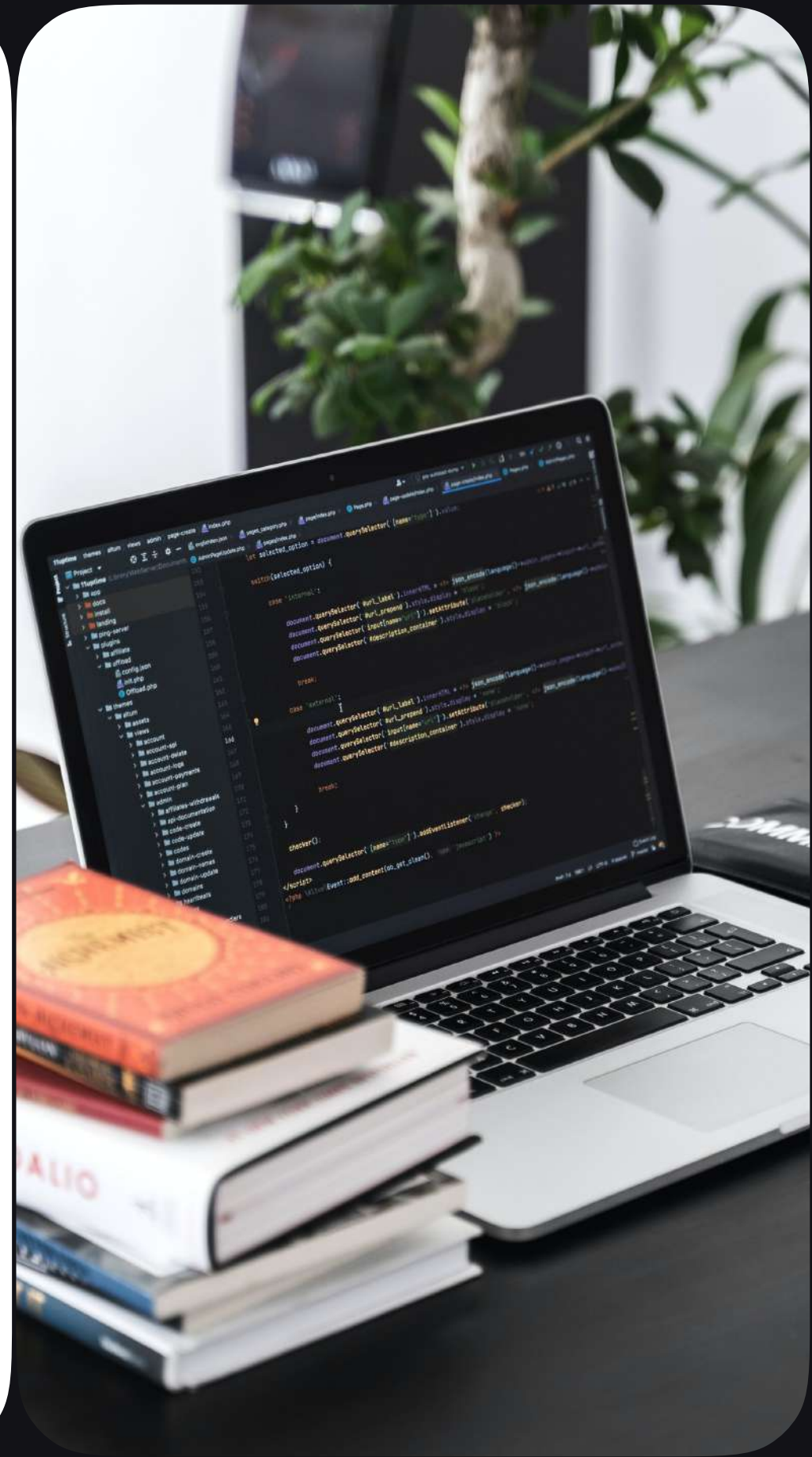


Модуль 4 Занятие 5

Формы. Flask-WTF



Формы

flask-wtf

CSRF

+

**Продолжаем создавать
новостной портал**

Формы

Формы являются неотъемлемой частью любого веб-приложения. Регистрация пользователей, добавление контента, комментарии — все это основано на работе с формами.

Create Account



Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna.

USER LOGIN

Welcome back

 Email or Username

 Password

☐ Remember

[Forgot password?](#)

LOGIN

WTForms

WTForms — это библиотека для работы с формами на Python. WTForms может работать с любыми фреймворками или даже без них.

Для работы с WTForms в Flask существует библиотека **Flask-WTF** — это интеграция фреймворка Flask и модуля WTForms, которая включает в себя защиту от межсайтовой подделки запросов, загрузку файлов, поддержку reCAPTCHA и много другое.

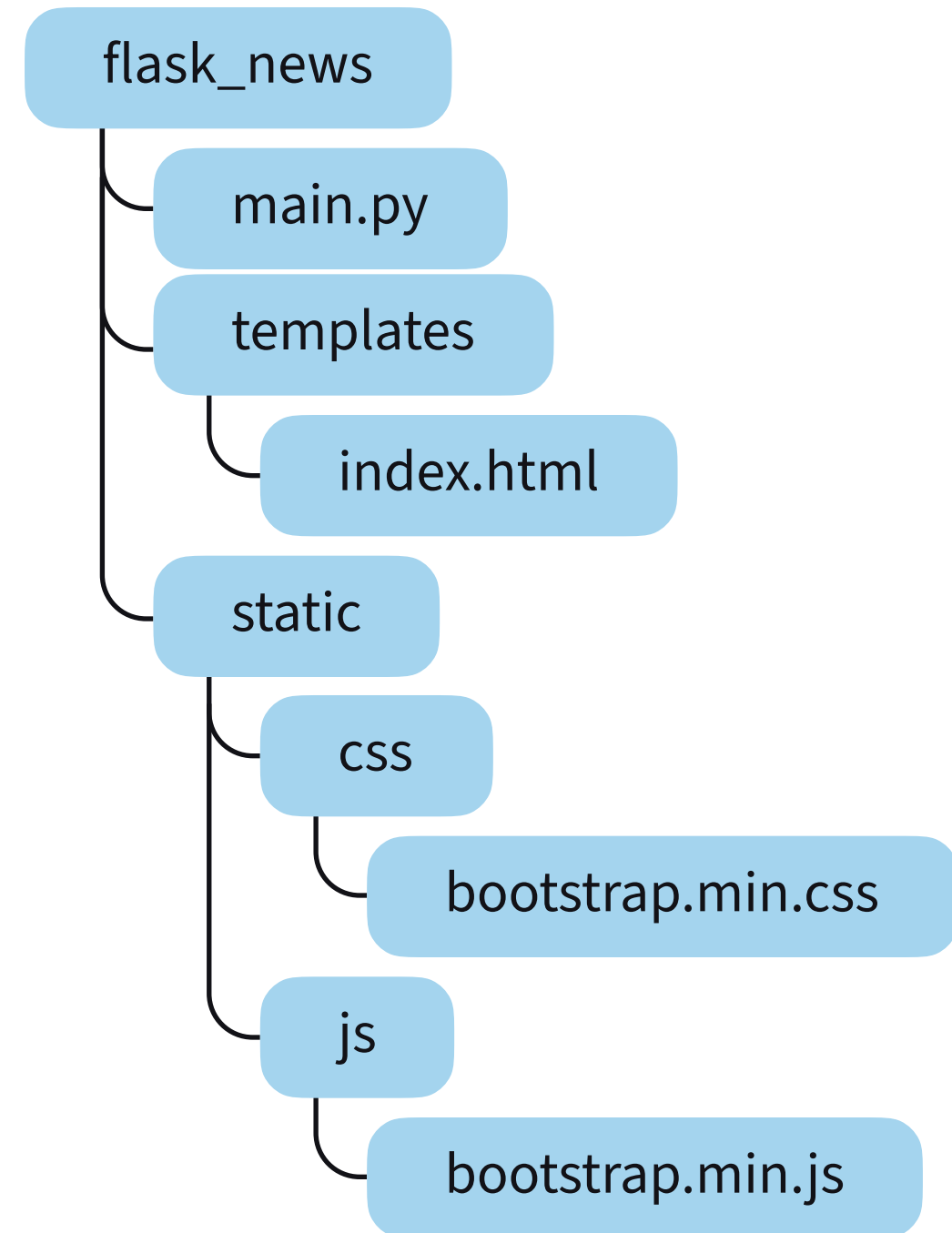
Для установки Flask-WTF в виртуальное окружение проекта выполните:

```
pip install flask-wtf
```

Структура проекта

Вернемся к проекту flask_app и поработаем с модулем flask-wtf.

Приведите структуру проекта к представленной. Папку со статикой можно взять из проекта flask_news.



Пример формы

```
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField
from wtforms.validators import DataRequired
```

```
class ExampleForm(FlaskForm):
    name = StringField('Имя', validators=[DataRequired()])
    submit = SubmitField('Отправить')
```

Для создания формы достаточно создать обычный Python класс, выполнив наследование от базового класса FlaskForm модуля Flask-WTF.

Каждое поле формы — это стандартный класс из модуля wtforms. Также можно указать валидатор, с помощью которого выполняется проверка корректности введенных данных.

Список полей

WTForms содержит большое количество различных полей, некоторые из них:

StringField

Текстовое поле

TextAreaField

Многострочное текстовое поле

PasswordField

Поле ввода пароля

EmailField

Поле ввода электронной почты

BooleanField

Чекбокс

SelectField

Выбор из списка

SubmitField

Кнопка отправки формы

Список валидаторов

WTForms содержит большое количество различных полей, некоторые из них:

DataRequired

Проверка ввода каких-либо данных в поле

Length

Проверка длины строки в поле

Email

Проверка email на валидность

URL

Проверка url на валидность

Optional

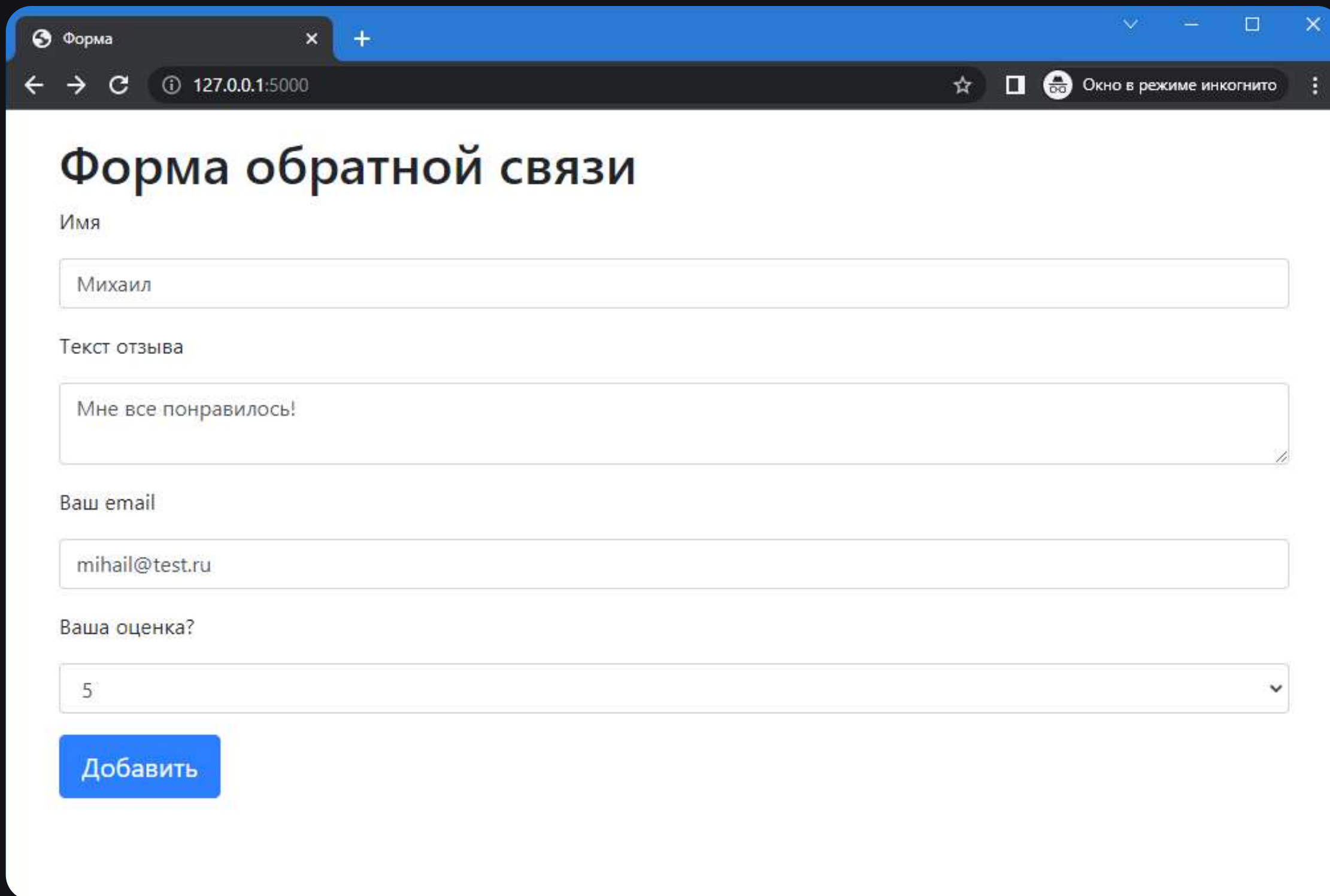
Делает поле ввода необязательным

CSRF

CSRF — вид атаки на посетителя сайта, использующий недостатки протокола HTTP. Если жертва заходит на сайт, созданный злоумышленником, то от ее лица может быть тайно отправлен запрос на другой сервер (например, на сервер платежной системы), осуществляющий некую вредоносную операцию (например, перевод денег на счёт злоумышленника).

Форма обратной связи

В качестве примера создадим форму обратной связи в проекте flask_app.



The screenshot shows a web browser window with a single tab titled 'Форма'. The address bar displays '127.0.0.1:5000' and indicates 'Окно в режиме инкогнито'. The page content features a heading 'Форма обратной связи' followed by four input fields: 'Имя' (containing 'Михаил'), 'Текст отзыва' (containing 'Мне все понравилось!'), 'Ваш email' (containing 'mihail@test.ru'), and 'Ваша оценка?' (a dropdown menu with '5' selected). A blue 'Добавить' button is positioned at the bottom left of the form area.

Код формы

main.py:

...

```
from flask_wtf import FlaskForm
```

```
from wtforms import StringField, SubmitField, TextAreaField, EmailField, SelectField
```

```
from wtforms.validators import DataRequired, Optional
```

...

```
class FeedbackForm(FlaskForm):
```

```
    name = StringField('Имя',  
                        validators=[DataRequired(message="Поле не должно быть пустым")])
```

```
    text = TextAreaField('Текст отзыва',  
                         validators=[DataRequired(message="Поле не должно быть пустым")])
```

```
    email = EmailField('Ваш email', validators=[Optional()])
```

```
    rating = SelectField('Ваша оценка?', choices=[1, 2, 3, 4, 5])
```

```
    submit = SubmitField('Добавить')
```

Передаем форму на страницу

main.py:

```
from flask import Flask, render_template
from flask_wtf import FlaskForm
from wtforms import StringField, TextAreaField, EmailField, SelectField, SubmitField
from wtforms.validators import DataRequired, Optional
```

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'SECRET_KEY'
```

```
class FeedbackForm(FlaskForm):
    ...
```

```
@app.route('/')
def index():
    form = FeedbackForm()
    return render_template('index.html', form=form)
```

У любого приложения есть определенные настройки. Настройки хранятся в объекте **config**, который можно рассматривать, как обычный словарь. Переменная конфигурации **'SECRET_KEY'** нужна для защиты от межсайтовой подделки запросов **CSRF**.

Для вывода формы создадим экземпляр класса формы и просто передадим его в шаблон с именем **form**.

Шаблон формы

index.html :

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="{{ url_for('static', filename='css/bootstrap.min.css') }}">
  <title>Форма</title>
</head>
<body>
  <div id="wrapper">
    <div class="container my-3">
      <form action="" method="POST" novalidate>
        {{ form.csrf_token }}
        <div class="form-group">
          <h1>Форма обратной связи</h1>
          <label>{{ form.name.label }}</label>
          {{ form.name(class="form-control") }}
        </div>
        ...
        {{ form.submit(class="btn btn-primary btn-lg") }}
      </form>
    </div>
  </div>
  <script src="{{ url_for('static', filename='js/bootstrap.min.js') }}"></script>
</body>
</html>
```

Создадим форму внутри контейнера `<form>...</form>`.

Поле `{{ form.csrf_token }}` в шаблоне создает специальный токен, который помещается в скрытое поле формы с именем `csrf_token` и передается из браузера обратно в функцию представление и, если этот токен не пройдет проверку как валидный, то запрос будет отклонен.

Шаблон формы

index.html :

...

```
<form action="" method="POST" novalidate>
```

```
  {{ form.csrf_token }}
```

```
  <div class="form-group">
```

```
    <h1>Форма обратной связи</h1>
```

```
    <label>{{ form.name.label }}</label>
```

```
    {{ form.name(class="form-control") }}
```

```
  </div>
```

```
  ...
```

```
  {{ form.submit(class="btn btn-primary btn-lg") }}
```

```
</form>
```

...

`form.<field_name>.label` — отображает заголовок поля, а `form.<field_name>` — само поле. Также можно передать дополнительные атрибуты HTML, например `class="form-control"` подключает стили формы из **bootstrap**.

Аналогично добавьте и другие поля формы.

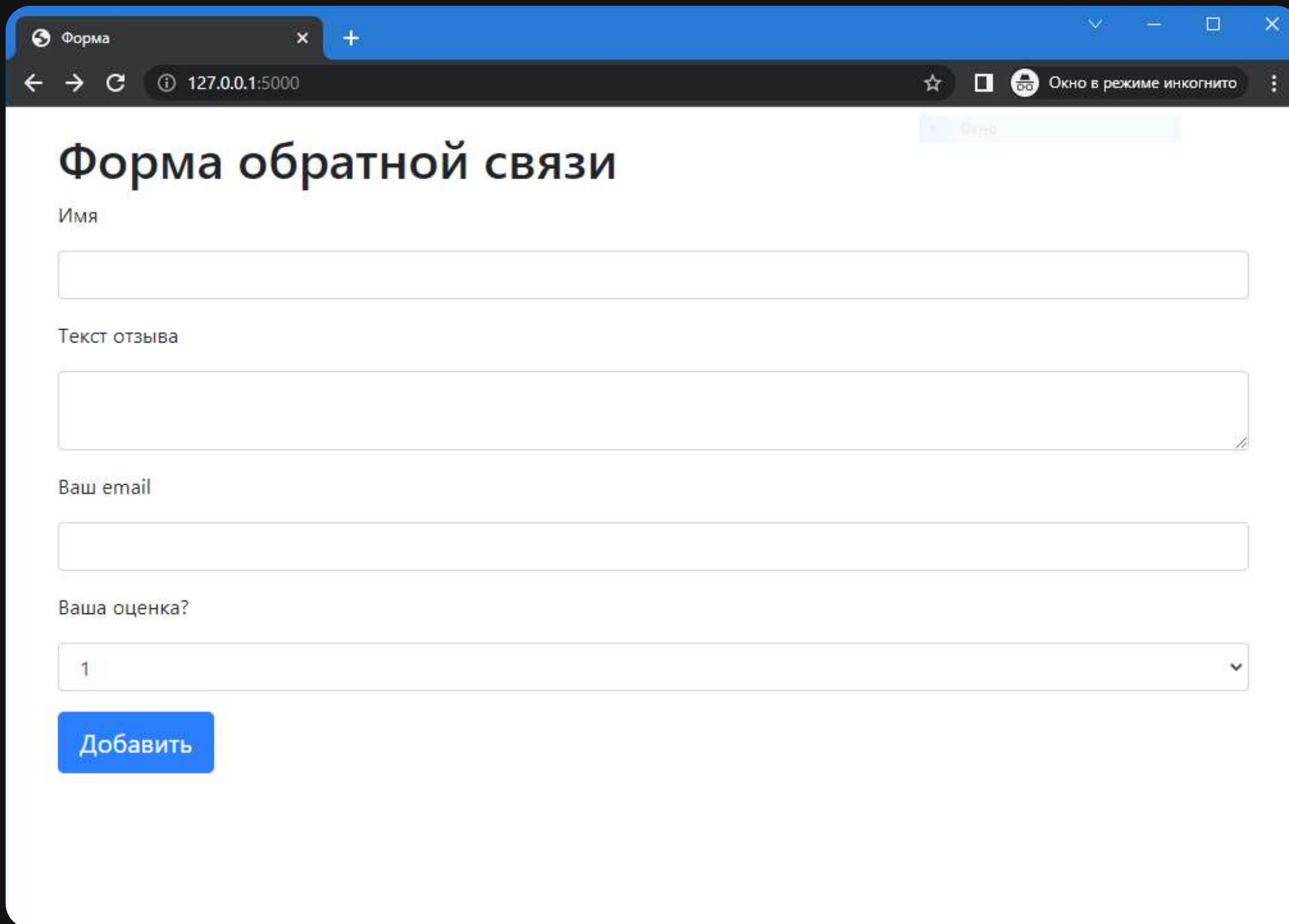
Все поля формы

...

```
<form action="" method="POST" novalidate>
  {{ form.csrf_token }}
  <div class="form-group">
    <h1>Форма обратной связи</h1>
    <label>{{ form.name.label }}</label>
    {{ form.name(class="form-control") }}
  </div>
  <div class="form-group">
    <label>{{ form.text.label }}</label>
    {{ form.text(class="form-control") }}
  </div>
  <div class="form-group">
    <label>{{ form.email.label }}</label>
    {{ form.email(class="form-control") }}
  </div>
  <div class="form-group">
    <label>{{ form.rating.label }}</label>
    {{ form.rating(class="form-control") }}
  </div>
  {{ form.submit(class="btn btn-primary btn-lg") }}
</form>
```

...

Результат



The screenshot shows a web browser window with a single tab titled 'Форма'. The address bar displays '127.0.0.1:5000' and indicates 'Окно в режиме инкогнито'. The page content is a feedback form with the title 'Форма обратной связи'. It contains four input fields: 'Имя' (Name), 'Текст отзыва' (Review text), 'Ваш email' (Your email), and 'Ваша оценка?' (Your rating?). The rating field is a dropdown menu currently showing '1'. A blue button labeled 'Добавить' (Add) is located at the bottom left of the form.

Форма

127.0.0.1:5000

Окно в режиме инкогнито

Форма обратной связи

Имя

Текст отзыва

Ваш email

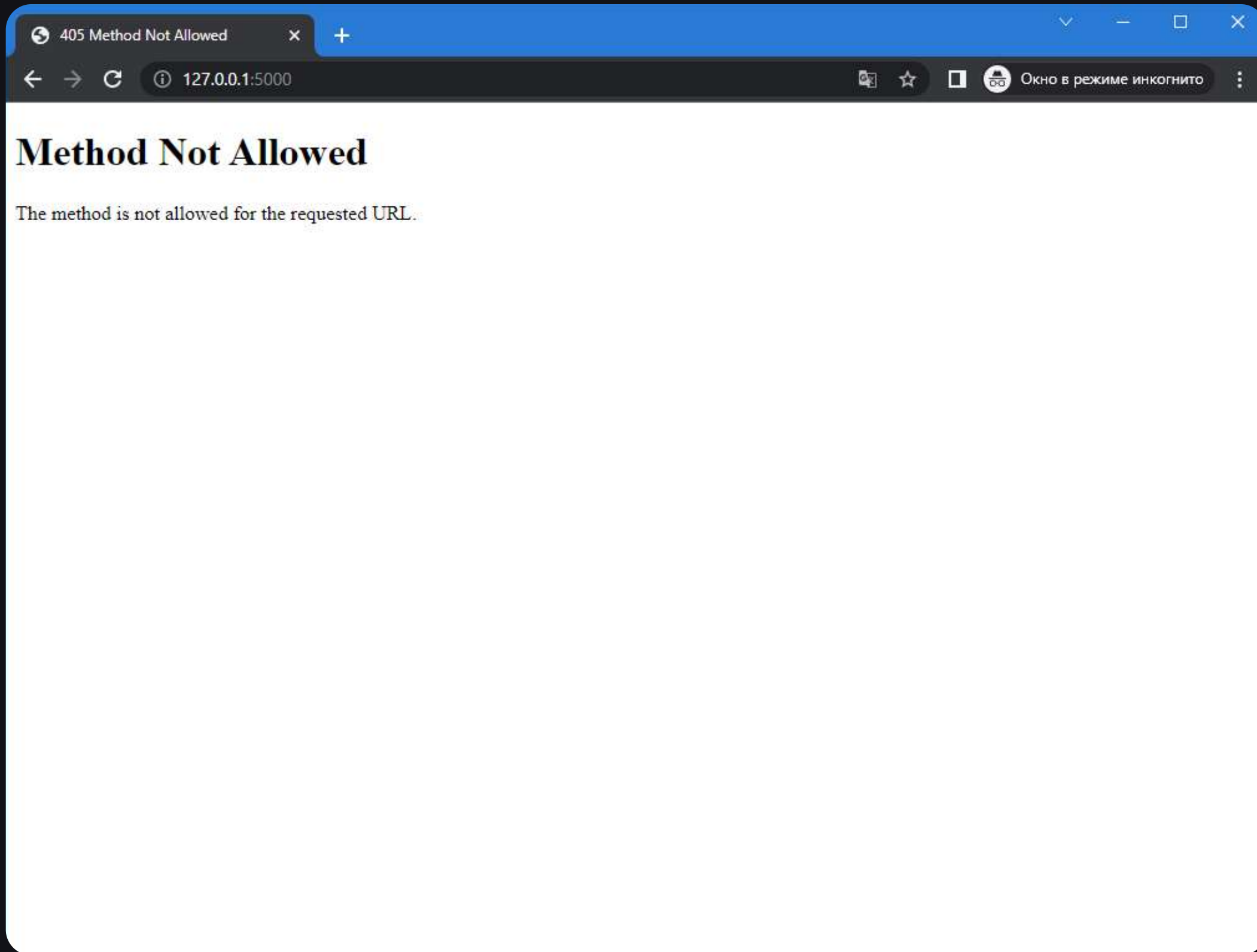
Ваша оценка?

1

Добавить

Запустите приложение, проверьте, что форма отображается, заполните поля формы и нажмите кнопку «Добавить».

Ошибка



В результате появится ошибка 405 Method Not Allowed — метод не поддерживается.

GET и POST

Методы **GET** и **POST** — самые распространённые методы для получения или отправки данных.

GET — метод для получения данных, например, для получения списка отзывов.



Информация передается
прямо в заголовке запроса:
GET /news/?name=Заголовок



Тело отсутствует

POST — метод для отправки данных, например, для отправки данных формы на сервер.



Заголовок:
POST /news/



Информация передается в теле
запроса:
?name=Заголовок

Обрабатываем данные формы

...

```
@app.route('/', methods=['GET', 'POST'])
```

```
def index():
```

```
    form = FeedbackForm()
```

```
    if form.validate_on_submit():
```

```
        name = form.name.data
```

```
        text = form.text.data
```

```
        email = form.email.data
```

```
        rating = form.rating.data
```

```
        print(name, text, email, rating)
```

```
        return redirect('/')
```

```
    return render_template('index.html', form=form)
```

Указываем разрешенные методы в параметрах декоратора @app.route.

Метод form.validate_on_submit() проверяет является ли запрос POST-запросом и запускает валидаторы для каждого поля. Если хотя бы один валидатор вернет ошибку — то метод вернет False. Получить доступ к данным формы можно, используя представленный синтаксис.

Вывод ошибок

...

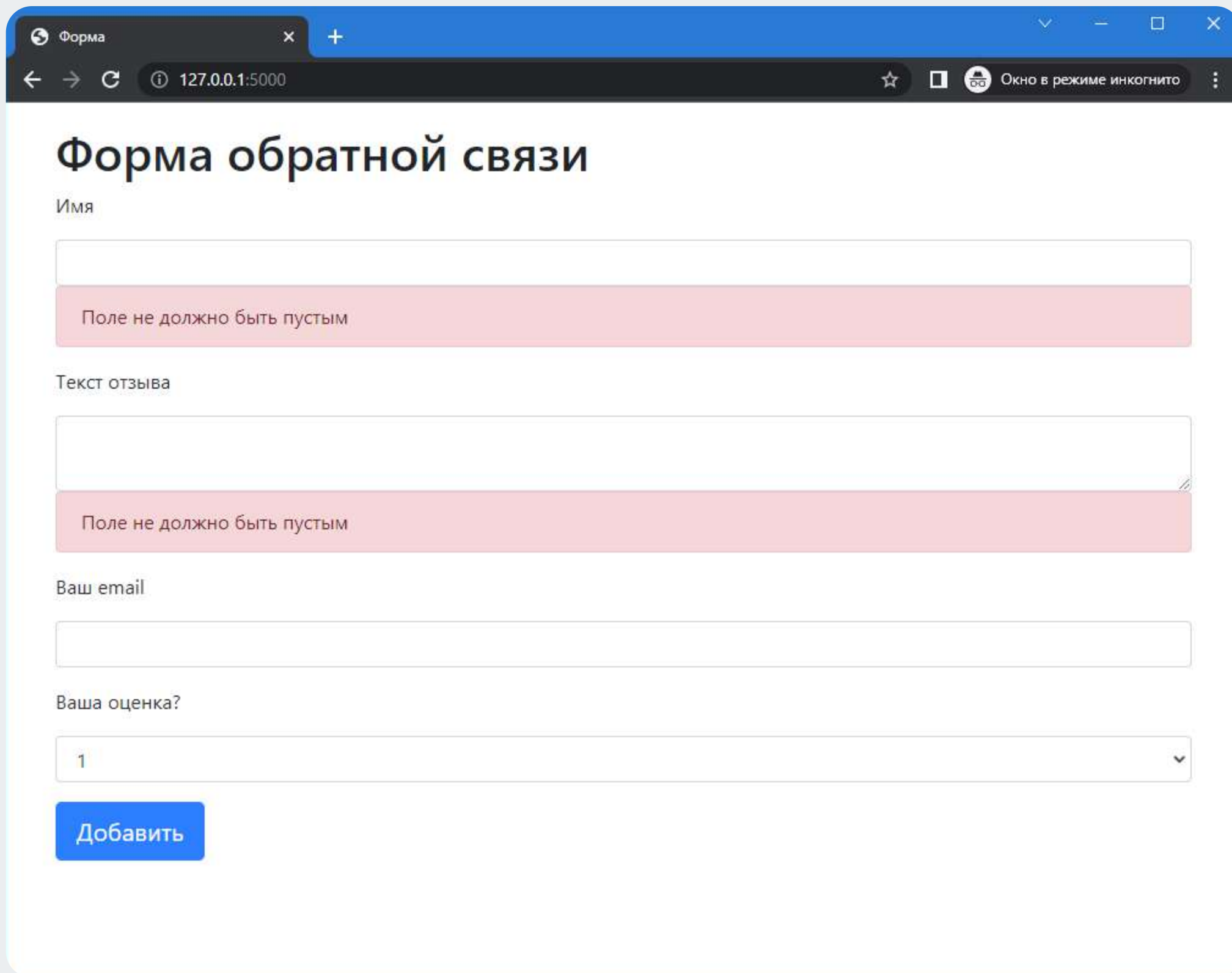
```
<form action="" method="POST" novalidate>
  {{ form.csrf_token }}
  <div class="form-group">
    <h1>Форма обратной связи</h1>
    <label>{{ form.name.label }}</label>
    {{ form.name(class="form-control") }}
    {% for error in form.name.errors %}
      <div class="alert alert-danger" role="alert">
        <span>{{ error }}</span>
      </div>
    {% endfor %}
  </div>
  ...
  {{ form.submit(class="btn btn-primary btn-lg") }}
</form>
```

...

При ошибках валидации для каждого поля будет доступен список ошибок `form.<field_name>.errors`. Выведем эти ошибки.

Аналогично добавьте вывод ошибок и в другие поля формы.

Результат



The screenshot shows a web browser window with the title 'Форма' and the address bar displaying '127.0.0.1:5000'. The page is titled 'Форма обратной связи' (Feedback Form). It contains four input fields: 'Имя' (Name), 'Текст отзыва' (Review text), 'Ваш email' (Your email), and 'Ваша оценка?' (Your rating?). The 'Имя' and 'Текст отзыва' fields are empty and have red error messages below them: 'Поле не должно быть пустым' (Field should not be empty). The 'Ваш email' field is empty. The 'Ваша оценка?' field is a dropdown menu with the value '1' selected. A blue button labeled 'Добавить' (Add) is at the bottom left.

Форма

127.0.0.1:5000

Окно в режиме инкогнито

Форма обратной связи

Имя

Поле не должно быть пустым

Текст отзыва

Поле не должно быть пустым

Ваш email

Ваша оценка?

1

Добавить

Теперь на странице можно видеть ошибки валидации.

ИТОГИ

✦ Для работы с формами в Flask используется библиотека Flask-WTF — это интеграция фреймворка Flask и модуля WTForms.

✦ Для создания формы достаточно создать обычный Python класс, выполнив наследование от базового класса FlaskForm, и использовать стандартные поля и валидаторы из wtforms.

✦ Для работы с формами в Flask используется библиотека Flask-WTF — это интеграция фреймворка Flask и модуля WTForms.

✦ Для создания формы достаточно создать обычный Python класс, выполнив наследование от базового класса FlaskForm, и использовать стандартные поля и валидаторы из wtforms.

Вернемся к flask_news

Проверим структуру проекта
и вспомним, на чем мы остановились.



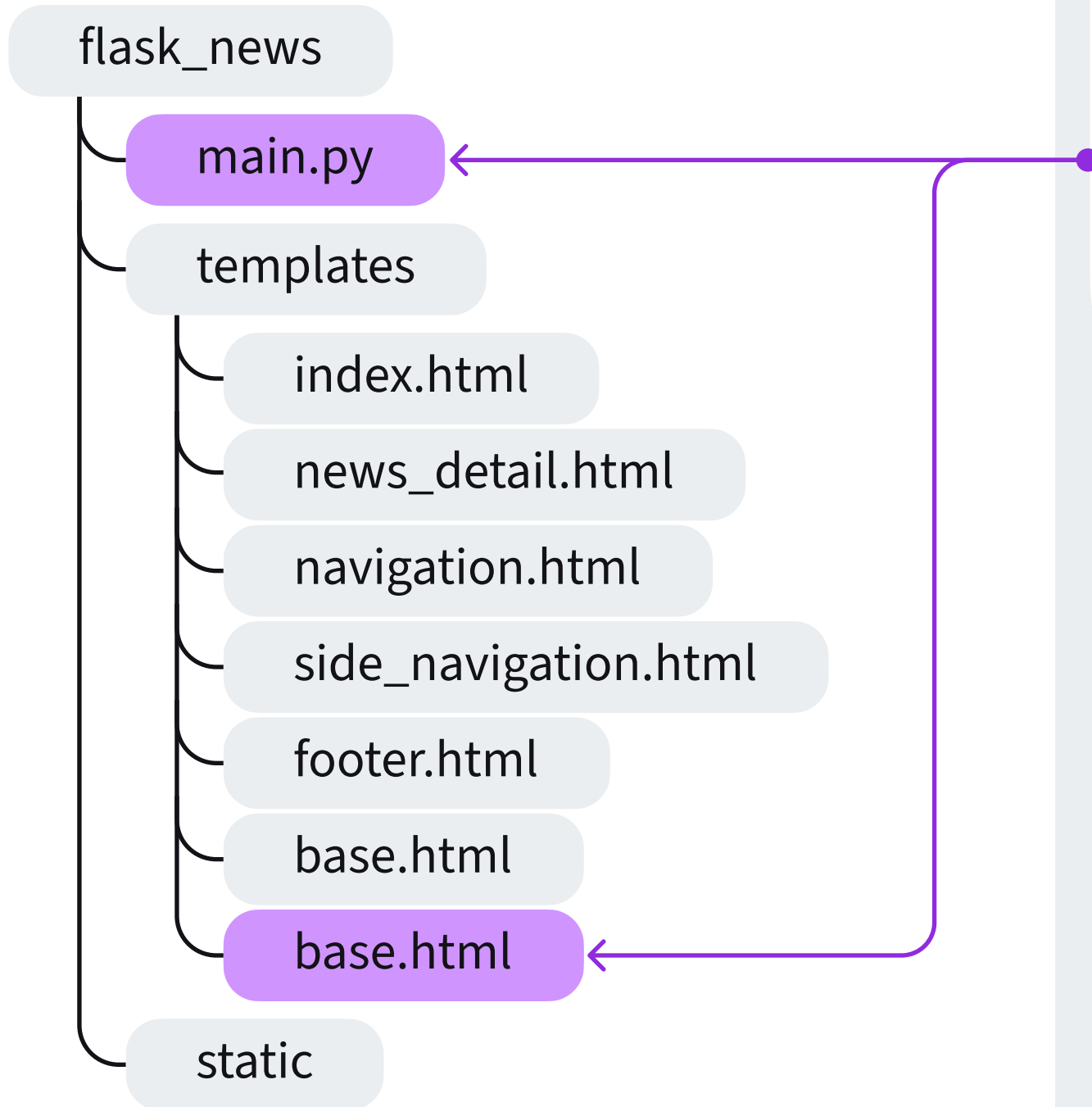
Проверьте работу проекта,
проверьте базовый и дочерние
шаблоны.



Проверьте работу всех ссылок.



Добавим новые файлы



Добавим в файл `main.py` код для создания формы, а также создадим новый шаблон `add_news.html` — шаблон для вывода формы.

Практика



Добавьте в приложение flask_news форму для добавления новости. Форма должна содержать два поля:



title (StringField) — обязательное поле длиной не более 256 символов.



text (TextAreaField) — обязательное поле.

Добавленная новость должна попадать в список news и отображаться на сайте.

Форма добавления новости

index.html :

```
...
from flask_wtf import FlaskForm
from wtforms import StringField, SubmitField, TextAreaField
from wtforms.validators import Length, DataRequired

...

class NewsForm(FlaskForm):
    title = StringField(
        'Название',
        validators=[DataRequired(message="Поле не должно быть пустым"),
                    Length(max=255, message='Введите заголовок длиной до 255 символов')]
    )
    text = TextAreaField(
        'Текст',
        validators=[DataRequired(message="Поле не должно быть пустым")]
    )
    submit = SubmitField('Добавить')
```

Шаблон формы

add_news.html:

```
{% extends "base.html" %}

{% block title %}
    Добавить новость
{% endblock title %}

{% block content %}

    <h1>Добавить новость</h1>

    <hr>

    <form method="POST" action="" novalidate>
        <!-- тут вывод формы -->
    </form>

{% endblock content %}
```

```
{{ form.csrf_token }}
<div class="form-group">
    <label >
        {{ form.title.label }}
    </label>
    {{ form.title(class="form-control") }}
</div>
{% for error in form.title.errors %}
    <div class="alert alert-danger" role="alert">
        <span>{{ error }}</span>
    </div>
{% endfor %}
<div class="form-group">
    <label >
        {{ form.text.label }}
    </label>
    {{ form.text(class="form-control", rows="5") }}
</div>
{% for error in form.text.errors %}
    <div class="alert alert-danger" role="alert">
        <span>{{ error }}</span>
    </div>
{% endfor %}
{{ form.submit(class="btn btn-primary btn-lg") }}
```

Обработка данных формы

main.py:

```
from flask import Flask, render_template, redirect, url_for, request
...

app = Flask(__name__)
app.config['SECRET_KEY'] = 'SECRET_KEY'

news = [...]

...

@app.route('/add_news', methods=['GET', 'POST'])
def add_news():
    form = NewsForm()
    if form.validate_on_submit():
        title = form.title.data
        text = form.text.data
        news.append({'title': title, 'text': text})
        return redirect(url_for('index'))
    return render_template('add_news.html',
                           form=form)
```

Результат

Добавить новость

127.0.0.1:5000/add_news

Окно в режиме инкогнито

НовостиНовостиДобавить новость

Добавить новость

Название

Важная новость!

Текст

Мы реализовали возможность добавлять новости всем желающим!

Добавить

Категории

Все новости

Категория

Категория

© 2023 Новости. Все права защищены.

Важная новость!

127.0.0.1:5000/news_detail/3

Окно в режиме инкогнито

НовостиНовостиДобавить новость

Важная новость!

Мы реализовали возможность добавлять новости всем желающим!

Категории

Все новости

Категория

Категория

© 2023 Новости. Все права защищены.