Код-ревью (из 20 ревью) от 01.12.2024 07:54

Файл: security.py

Строка 12

- **Комментарий:**

Отсутствует аннотация типов для параметра функции и возвращаемого значения. Рекомендуется добавить аннотации для улучшения читаемости и поддержки кода.

- **Текущий код:**

```
# token. This method is passed the identity of who the token is being
# created for, and must return data that is json serializable
# @jwt.user_claims_loader
def add_claims_to_access_token(identity):
    return {
```

Строка 16

- **Комментарий:**

Использование точечного доступа к атрибутам объекта (identity.id) может привести к ошибке AttributeError, если атрибут отсутствует. Рассмотрите возможность использования getattr с значением по умолчанию или проверки наличия атрибута.

- **Текущий код:**

```
def add_claims_to_access_token(identity):
    return {
        'user_id': identity.id,
        'username': identity.username,
```

Строка 18

- **Комментарий:**

Аналогично предыдущему комментарию, identity.username может вызвать AttributeError. Рекомендуется добавить проверку или использовать getattr.

- **Текущий код:**

```
'user_id': identity.id,
        'username': identity.username,
        'roles': [role.name for role in identity.roles]
```

Строка 20

- **Комментарий:**

Использование спискового включения для получения имен ролей может быть неэффективным при большом количестве ролей. Рассмотрите возможность использования генераторного выражения для экономии памяти.

- **Текущий код:**

```
'username': identity.username,
        'roles': [role.name for role in identity.roles]
    }
```

Строка 24

- **Комментарий:**

Функция имеет слишком общий название. Рассмотрите возможность использования более специфического названия, например, handle_missing_jwt.

- **Текущий код:**

```
}


def no_jwt_for_protected_endpoint(error_message):
    return jsonify({'success': False, 'full_messages': [error_message]})
```

Строка 27

- **Комментарий:**

  Использование 'full_messages' в ответе может быть нестандартным. Рассмотрите возможность использования 'message' или 'error' для более ясного понимания.

- **Текущий код:**

```
def no_jwt_for_protected_endpoint(error_message):
    return jsonify({'success': False, 'full_messages': [error_message]})
```

Строка 29

- **Комментарий:**

  Пустая строка в конце функции не нужна. Удалите ее для улучшения читаемости.

- **Текущий код:**

```
return jsonify({'success': False, 'full_messages': [error_message]})


# This is called on request, when the user accesses a restricted endpoint with a jwt token
```

Строка 37

- **Комментарий:**

  Отсутствует обработка случая отсутствия пользователя. Рекомендуется добавить проверку на None и обработку исключений.

- **Текущий код:**

```
# that you provided in identity_loader(), that value was issued to the user, now you get it back,
# then we have the id back :)
def user_loader(user_id):
    # the user will be now available in current_user
    return User.query.get(user_id)
```

Строка 41

- **Комментарий:**

  Использование User.query.get(user_id) может быть неэффективным при работе с большими базами данных. Рассмотрите возможность использования User.query.filter_by(id=user_id).first() для более явного указания намерений.

- **Текущий код:**

```
    # the user will be now available in current_user
        return User.query.get(user_id)


    # This is called when a jwt token is gonna be created,
```

Строка 45

- **Комментарий:**

  Функция identity_loader принимает объект user, но не проверяет его наличие или тип. Рекомендуется добавить проверку на None и убедиться, что у объекта есть атрибут id.

- **Текущий код:**

```
    # This is called when a jwt token is gonna be created,
    # you have to pass a json serializable object that will be used as the id of the user(identity)
    def identity_loader(user):
        return user.id
```

Строка 48

- **Комментарий:**

  Возвращается user.id без проверки. Если user.id может быть None или вызывать исключение, добавьте обработку таких случаев.

- **Текущий код:**

```
    def identity_loader(user):
        return user.id
```

Строка 51

- **Комментарий:**

  Функция не принимает никаких аргументов и всегда возвращает одинаковый результат. Возможно, стоит добавить параметры для более гибкого использования и возвращения различных сообщений в зависимости от ситуации.

- **Текущий код:**

```
def token_revoked():
    return jsonify({'success': False, 'full_messages': ['Revoked token']})
```

Строка 56

- **Комментарий:**

  Функция использует необъявленную функцию jsonify. Рекомендуется импортировать jsonify из flask: from flask import jsonify.

- **Текущий код:**

```
return jsonify({'success': False, 'full_messages': ['Revoked token']})


def invalid_token_loader(error_message):
```

Строка 58

- **Комментарий:**

  Пустая строка между определением функции и ее телом не требуется. Удалите пустую строку для улучшения читаемости кода.

- **Текущий код:**

```
def invalid_token_loader(error_message):
    return jsonify({'success': False, 'full_messages': [error_message]})
```

Строка 60

- **Комментарий:**

  Использование списка для одного сообщения может быть избыточным. Если сообщений всегда будет один, используйте строку вместо списка: return jsonify({'success': False, 'full_message': error_message})

- **Текущий код:**

```
def invalid_token_loader(error_message):
    return jsonify({'success': False, 'full_messages': [error_message]})
```

Строка 78

- **Комментарий:**

  Использование прямого доступа к элементам словаря может вызвать KeyError, если ключ отсутствует. Рассмотрите использование метода get с значением по умолчанию.

- **Текущий код:**

```
@jwt.expired_token_loader
def valid_but_expired_token(expired_token):
    token_type = expired_token['type']
    return jsonify({
```

Строка 80

- **Комментарий:**

  Функция возвращает JSON-ответ и код состояния, но не импортирует jsonify. Хотя это не считается ошибкой, убедитесь, что jsonify импортирован в начале файла.

- **Текущий код:**

```
token_type = expired_token['type']
    return jsonify({
        'success': False,
        'full_messages': [
```

Строка 83

- **Комментарий:**

  Использование списка для одного сообщения избыточно. Рассмотрите возможность использования строки вместо списка.

- **Текущий код:**

```
'success': False,
        'full_messages': [
            'Token expired'
        ]
```

Строка 91

- **Комментарий:**

  Список допустимых расширений лучше хранить в виде множества (set) для более быстрого поиска.

- **Текущий код:**

```
def validate_file_upload(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ['png', 'jpeg', 'jpg']
```

Строка 92

- **Комментарий:**

  Использование обратного слэша для переноса строк не рекомендуется. Рассмотрите использование круглых скобок для улучшения читаемости.

- **Текущий код:**

```
def validate_file_upload(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ['png', 'jpeg', 'jpg']
```