Тестовое задание

После проверки задания мы обязательно прокомментируем ваше решение. Код тестового задания не будет использован в коммерческих целях.

## Описание

[stripe.com/docs](http://stripe.com/docs) - платёжная система с подробным API и бесплатным тестовым режимом для имитации и тестирования платежей. С помощью python библиотеки stripe можно удобно создавать платежные формы разных видов, сохранять данные клиента, и реализовывать прочие платежные функции.

Мы предлагаем вам познакомиться с этой прекрасной платежной системой, реализовав простой **сервер** с одной html страничкой, который общается со **Stripe** и создает платёжные формы для товаров.

Для решения нужно использовать **Django**. Решение бонусных задач даст вам возможность прокачаться и показать свои умения, но это не обязательно.

## Задание

Реализовать Django + Stripe API бэкенд со следующим функционалом и условиями:

* Django Модель Item с полями (name, description, price)
* API с двумя методами:
  + **GET** /buy/{id}, c помощью которого можно получить Stripe Session Id для оплаты выбранного Item. При выполнении этого метода c бэкенда с помощью python библиотеки **stripe** должен выполняться запрос ***stripe.checkout.Session.create(...)*** и полученный session.id выдаваться в результате запроса
  + **GET** /item/{id}, c помощью которого можно получить простейшую HTML страницу, на которой будет информация о выбранном Item и кнопка **Buy**. По нажатию на кнопку **Buy** должен происходить запрос на /buy/{id}, получение session\_id и далее с помощью JS библиотеки **Stripe** происходить редирект на Checkout форму ***stripe.redirectToCheckout(sessionId=session\_id)***
  + Пример реализации можно посмотреть в пунктах 1-3 [тут](https://stripe.com/docs/payments/accept-a-payment?integration=checkout)
* Залить решение на Github, описать запуск в Readme.md
* Опубликовать свое решение чтобы его можно было быстро и легко протестировать.
* Решения доступные только в виде кода на Github получат низкий приоритет при проверке.

Бонусные задачи:

* Запуск используя Docker
* Использование environment variables
* Просмотр Django Моделей в Django Admin панели
* Запуск приложения на удаленном сервере, доступном для тестирования
* Модель Order, в которой можно объединить несколько Item и сделать платёж в Stripe на содержимое Order c общей стоимостью всех Items
* Модели Discount, Tax, которые можно прикрепить к модели Order и связать с соответствующими атрибутами при создании платежа в Stripe - в таком случае они корректно отображаются в Stripe Checkout форме.
* Добавить поле Item.currency, создать 2 Stripe Keypair на две разные валюты и в зависимости от валюты выбранного товара предлагать оплату в соответствующей валюте
* Реализовать не Stripe Session, а Stripe Payment Intent.

Пример

API метод для получения HTML c кнопкой на платежную форму от Stripe для Item с id=1:

curl -X GET [http://localhost:8000/](http://localhost:8000/posts)item/1

Результат - HTML c инфой и кнопкой:  
<html>

<head>

<title>Buy Item 1</title>

</head>

<body>

<h1>Item 1</h1>

<p>Description of Item 1</p>  
 <p>1111</p>

<button *id*="buy-button">Buy</button>

<script *type*="text/javascript">

var stripe = Stripe('pk\_test\_a9nwZVa5O7b0xz3lxl318KSU00x1L9ZWsF');

var buyButton = document.getElementById(buy-button');

buyButton.addEventListener('click', function() {

// Create a new Checkout Session using the server-side endpoint

// Redirect to Stripe Session Checkout

fetch('/buy/1', {method: 'GET'})

.then(response => return response.json())

.then(session => stripe.redirectToCheckout({ sessionId: session.id }))

});

</script>

</body>

</html>

iirp-thqf-awit-rfgf-fibl