Техническое задание на разработку PROXY

Общая информация

Нужно создать сайт для продажи прокси - серверов. При создании сервиса функционал в основном копируем с сайта донора https://nsocks.net/ и во многом ориентируемся на этот ресурс.
Вход в ЛК для изучения:
Login: Socks377
Parol: Ziny37212602$

Сайт должен обеспечивать клиентов надежными и безопасными прокси для различных целей:

- предоставление клиентам удобной платформы для выбора, заказа и управления прокси
- обеспечение администраторам инструментов для управления заказами и клиентами
- обеспечение безопасности и производительности сайта.

Дизайн должен быть современным и привлекательным, согласованным с тематикой сайта. Сайт должен быть адаптивным для мобильных устройств и кроссбраузерным. Навигация должна быть легкой и интуитивно понятной.

Макет должен включать следующие страницы:

- Главная страница - Main
- Страница активных (купленных) прокси в ЛК - My Proxies
- Страница платежей в ЛК - Payments
- Страница заказа и оплаты прокси (корзина) ЛК - Cart
- Страница истории заказов ЛК - History
- Страница новостей - News
- Страница правил и условий - Terms
- Страница поддержки - Support
- Административная панель для управления заказами, клиентами и прокси - серверами.

Функциональность

- Регистрация и аутентификация пользователей.
- Система фильтрации и выбора прокси по параметрам (геолокация, скорость, тип прокси и другие).

- Корзина заказов
- Внутренний счет с возможностью пополнения через Stripe
- Возможность оформления и оплаты заказов с внутреннего счета
- Личный кабинет для клиентов с историей заказов и возможностью изменения настроек.
- Чат-поддержка для вопросов и помощи клиентам.
- Панель администратора для управления прокси - серверами, заказами, клиентами, аналитикой и настройками сайта.

Технологии

Веб-сервер: Nginx или Apache.
Язык программирования: PHP, фреймворк Laravel.
Система управления базами данных: MySQL.
Фронтенд: HTML5, CSS3, JavaScript (Vue.js).

Задача 1: Верстка макета

По созданному макету от дизайнера необходимо выполнить верстку страниц будущего сайта. За основу страниц для верстки был взят сайт https://nsocks.net

Задача 2: Подготовка приложения

Нужно настроить фреймворк Laravel для работы над приложением. Настройка включает в себя:

- добавление нужных пакетов Composer
- интеграция готовой верстки на основе макета во фронт приложения
- установка и настройка админ панели приложения, интеграция шаблона административной панели - Laravel Nova
- деплой приложения на тестовый сервер разработки и перенос актуальной копии БД

Задача 3: Создание структуры БД

Нужно создать определенную структуру БД приложения. Должны существовать такие таблицы БД:

- user
- admin
- proxy
- isp
- country

- flag
- city
- proxy_type
- proxy_user
- proxy_country
- payment
- log
- ticket
- pages
- news

Задача 4: Настройка сервера и окружения

Для правильной работы с прокси нужно реализовать поддержку генерации новых прокси, отслеживание действующих и постоянный контроль их состояния. Данная работа будет выполнена системным администратором на основе отдельного технического задания.
Ссылка на ТЗ

Задача 5: Админпанель: модуль пользователей и ролей

Нужно создать модуль управления пользователями в админпанели. Для сохранения данных о пользователях нужно создать CRUD с такими данными о пользователе:

- username (varchar)
- электронная почта (varchar)
- пароль (varchar)
- статус в системе (int)

В административной панели должна быть возможность добавить пользователя без его регистрации. Для этого вводится его почта и пароль. Поле пароля заполняется либо самим админом, либо генерируется случайный пароль и сохраняется в системе. В админпанели должна быть возможность легко заменить пароль, а также установить статус пользователя в системе. В модуле есть такие статусы пользователей:

- guest (гость)
- authenticated (залогиненный)
- banned (забанен и не может работать в ЛК)

Также нужно создать модуль управления ролями пользователей.
В системе присутствуют такие роли:

- guest
- user

- manager
- admin

Не аутентифицированные пользователи имеют роль guest. По умолчанию во время регистрации пользователю присваивается роль user. Роль admin является наивысшей по привилегиям и имеет наибольшие права в системе. Роль manager может присвоить любому пользователю только admin.

Задача 6: Настройка авторизации, регистрации, сброса пароля

При регистрации нового пользователя нужно использовать поля username, почты и пароля. Почту подтверждаем по переходу по ссылке с токеном подтверждения (ссылка отправляется на введенную пользователем почту при регистрации).

То же касается и сброса пароля - при запросе сброса отправляем на почту ссылку на сброс, где на отдельной странице показываем поле ввода и подтверждения нового пароля. При смене пароля записи данных входа пользователя в БД изменяются на новые, пользователю отображается уведомление что его данные были изменены.

Задача 7: Настройка локализации и языков

Фронт приложения должен работать пока что только на одном языке:

- en

Но должна быть возможность со временем добавить другие языки, необходимо настроить приложение так чтобы добавить новый язык не было большой проблемой. Необходима установка компонента локализации во фреймворк и удобная работа с урл языковых версий. Язык в урл должен отображаться сразу после названия домена. По умолчанию дефолтный язык - en и он не отображается в адресной строке при переходе на английскую версию. Когда будут добавлены другие языки, то они должны отображаться в адресной строке при смене языковой версии сайта.

Остальные языки должны легко добавляться путем копирования язкового файла и перевода на нужный язык. В коде весь контент (кроме того что выводится из БД) должен выводиться из языковых переменных во избежание проблем с добавлением новых языков. Структура таблицы pages и news должна иметь колонки title_en, text_en, для получения из базы контента на выбранном пользователем языке.

Задача 8: Создание полной ноды для пополнения внутреннего баланса

Нужно на хостинге, где расположены прокси либо на другом хостинге, который наилучше подойдет для этого развернуть собственную полную ноду, то есть выделенный сервер с необходимым ПО для доступа к блокчейну, синхронизации и

хранения истории транзакций. Предполагается пополнение двумя видами криптовалют:

- BitCoin
- LiteCoin

Нужно сделать возможность пополнения внутреннего баланса с помощью своей криптовалютной ноды. Пользователи должны иметь возможность пополнять кошельки нашей ноды, при подтверждении транзакций (количество подтверждений зависит от валюты пополнения) внутренний счет пользователя пополняется на переведенную им сумму.

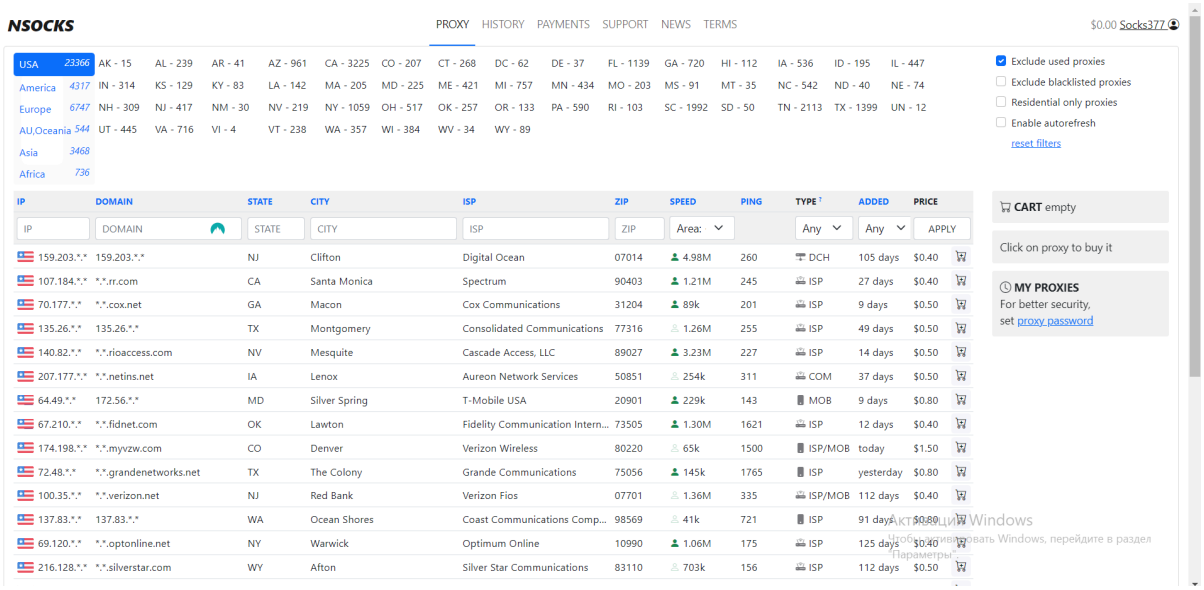
Для конвертации пополненной криптовалюты во внутренний баланс нужно подключить АПИ сервиса https://coinmarketcap.com/. Во время операции пополнения мы отправляем запрос на сервис и получаем актуальный курс BTC -> USD и выполняем пополнение баланса на полученную сумму в USD. Если сумма пополнения USD идет с остатком, то округляем сумму до целого числа в большую сторону. Например, юзер пополнил в биткоинах на 4.45 USD или 4.55 USD, начисляем ему 5 USD.

На странице пополнения счета у пользователя должна быть возможность сгенерировать кошельки для пополнения через Bitcoin и LiteCoin. Эти кошельки сохраняются в его ЛК и в дальнейшем не изменяются, через них он может пополнять нашу ноду.

Задача 9: Создание вкладки “Proxies” ЛК

При переходе на вкладку “Proxies” пользователю должно отобразиться:

- список список количества доступных прокси по странам
- список всех прокси, который доступен для покупки по всем странам
- фильтры для поиска нужных прокси по параметрам (страна, провайдер, тип…)
- фильтры - чекбоксы
- пагинация прокси, работающая без перезагрузки страницы



Список доступных прокси по странам должен отображать количество прокси в конкретной стране. Список стран выводится в форме двузначного кода страны и количеством прокси в этой стране.

По умолчанию при загрузке активной вкладкой отображается страна USA (слева) с доступным количеством прокси - серверов по штатам (справа). При переключении вкладки на другую страну или континент данные по регионам и количествам прокси в них меняются.

Справа должны отобразиться регионы, относящиеся к выбранной стране (континенту) слева. Также меняется и общий список прокси с пагинацией - в зависимости от выбранной страны (континента). Фильтрация должна работать так, что при изменении одного из фильтров все данные рендерятся заново и отображается актуальная информация без перезагрузки.

Для сохранения данных о прокси нужно создать CRUD с такими данными:

- ip (varchar)
- domain (varchar)
- state (varchar)
- speed (varchar)
- ping (int)
- add_date (date)
- price (decimal 8,2)
- type_id (int)
- flag_id (int)
- country_code (varchar)
- city_id (int)
- isp_name (varchar)
- isp_id (int)
- zip_code (int)
- status (int)

Таблица “proxy” предусматривает создание связей таблиц. Например, нам нужно создать таблицы country и city, где будут списки стран и связанных по айди с ними городов. В таблице proxy колонка country_code связана с таблицей country по foreign key. То же касается таблицы proxy_type, которая связана через колонку type_id.

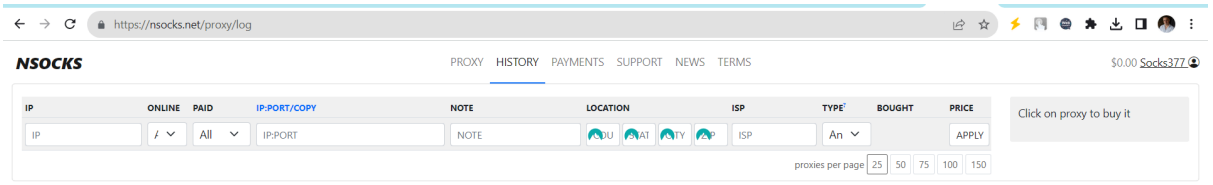
Таблица flag, где хранятся изображения флагов стран, также связана про колонке flag_id. Возможно логично хранить путь к флагу страны в таблице country. В общем изучаем структуру данных для сохранения и прорабатываем все внешние связи с таблицами по ID.

Задача 10: Создание корзины

Повторить функционал nsocks.net.

Задача 11: Создание вкладки “History” ЛК

На этой вкладке должны отображаться все прокси, которые ранее покупались пользователем за любой период. Также должны быть фильтры для поиска нужных прокси в истории.



Задача 12: Создание вкладки “Мои прокси” ЛК

Пользователи должны иметь возможность просматривать список активных (купленных) прокси в аккаунте на данный момент. Также на этой странице должна быть возможность продлить аренду прокси на требуемый период времени.

Задача 13: Создание вкладки “Мои платежи” ЛК

Пользователи должны иметь возможность просматривать список произведенных платежей за аренду прокси. Также делаем доступными фильтры для поиска платежа по айди, сумме, дате платежа.

Задача 14: Создание вкладки “Мой профиль” ЛК

Во вкладке “Мой профиль” пользователь может сменить свой пароль, введя старый пароль и новый дважды. После смены пароля его данные перезаписываются и впредь он входит по новому паролю. Также должна быть вкладка “Изменить Socks5 login:pass”, где пользователь может сгенерировать пароль или добавить свой и использовать для прокси.

Задача 15: Подключение шлюза отправки смс - сообщений

Для отправки сообщений планируется использовать сервис Postmarkapp.com
Нужно будет создать аккаунт на корпоративную почту сервиса и настроить отправку уведомлений пользователям. Смс будет использоваться для двухфакторной аутентификации и регистрации пользователя.

Задача 16: Админ панель: управление прокси

В админпанели нужно создать страницу для управления прокси. Прежде всего мы должны видеть все прокси, созданные (сгенерированные) в нашей системе.

В админ панели должны точно видеть:

- статус прокси (действующий, в черном списке и пр.)
- вся информация о прокси, которую видит пользователь
- сколько раз он был использован пользователями
- кто им пользуется сейчас (если он куплен)
- срок действия прокси для пользователя и когда истекает
- возможность изменить цену для этого прокси индивидуально

В админпанели также нужно создать страницу настроек цены для всех прокси, которые мы генерируем. Цена на прокси будет устанавливаться автоматически при его генерации в системе. На установку цены будет влиять 2 фактора:

- тип прокси (мобильный, резидентный, серверный)
- срок его жизни (создан сегодня, вчера, 3 дня назад, неделю назад, месяц назад)

В админпанели нужно создать страницу настроек цены за прокси по их типу и продолжительности жизни. Это должна быть сетка цен, разбитая на 3 типа прокси и

несколько типов периодов жизни прокси. Цены на прокси нереальные, просто показано для примера.

Тип/Когда создан	Мобильный	Резидентный	Серверный
сегодня	2$	3$	5$
вчера	1$	0.8$	0.6$
до 1 недели	0.8$	0.6$	0.4$
до 1 месяца	0.5$	0.3$	0.2$

То есть цена за прокси у нас должна меняться в зависимости от периода его жизни. Если с типом все определенно понятно, срок жизни прокси изменяет его цену со временем. Поэтому нам нужен cron, который будет бегать каждый день и делать изменения цен прокси согласно этим настройкам цен из админки.

Задача 17: Тариф “Безлимит на сутки”

В рамках нашего проекта должна быть создана для пользователя возможность купить безлимит пользования любыми прокси на сутки. То есть, нужно будет создать вкладку в админке, где мы указываем цену безлимита, к примеру 70$. При покупке прокси мы должны предложить пользователю возможность (кроме покупки на сутки) также купить и безлимит на сутки. Это может быть отображено в корзине, либо как то в всплывающем окне. Эта возможность у пользователя должна быть всегда и везде в ЛК.

После оплаты безлимита пользователь может неограниченно пользоваться любыми прокси (получать к ним доступ, ссылку для подключения - при условии что у него установлен логин - пароль). Мы не ограничиваем пользователя ни по количеству прокси, ни по типу. После истечения срока безлимита доступ к заказанным прокси должен быть ограничен и пользователю дана возможность возобновить безлимит, оплатив его еще раз.

Для учета безлимитов предлагаю использовать отдельную таблицу **user_unlim_proxies**, по типу **user_proxies,** чтобы не путались данные.

Задача 18: Построение АПИ в веб - приложении и на хостинге

Для получения актуальных прокси веб - приложение должно отправлять АПИ - запрос на веб - сервер на хостинге прокси - серверов. Со стороны веб - приложения нужно создать АПИ, которое будет отправлять запросы и получать ответы от другого АПИ, которое будет расположено на хостинге серверов. Основные действия - команды для АПИ:

- получение актуальных данных о имеющихся прокси - серверах на хостинге (их список, количество, какие docker - контейнеры из них запущены и работают в данный момент). В общем советуемся с DevOps и получаем максимально возможную информацию о прокси - серверах на хостинге VPS. Ответ должен приходить в виде JSON - массива.
- назначение логина и пароля для конкретного прокси и запуск докер - контейнера с прокси (его включение и доступность для пользователя). Ответ должен приходить в виде JSON - массива с данными об успешности ил неуспешности команды.
- остановка докер - контейнера по событию завершения срока аренды. Ответ должен приходить в виде JSON - массива с данными об успешности ил неуспешности команды.

Веб - сервер, который расположен на хостинге прокси - серверов, получая запрос извне (от веб - приложения), должен запустить команду на получение данных о имеющихся прокси - серверах и выдать по АПИ ответ в виде строки JSON. В строке должны быть структурированные данные о прокси - серверах приблизительно в таком виде:


```
[{
   “name”: “Proxy_1”,
   “domain”: “121.45.345”,
   “ip”: “121.45.345”,
   “port”: “8080”,
   “speed”: “80”,
   “isp”: “Magyar Telecom”,
   “country”: “hr”,
   “city”: “Budapest”,
   “ping”: “100”,
  “created_date”: “2023-11-06”,
  “status”: 1
},
{
   “name”: “Proxy_2”,
   “domain”: “151.37.202”,
   “ip”: “151.37.202”,
   “port”: “2020”,
   “speed”: “100”,
   “isp”: “Union Provider”,
   “country”: “us”,
   “city”: “New York”,
  “ping”: “200”,
  “created_date”: “2023-11-04”,
  “status”: 0

},
]
```

И так мы получаем данные о всех прокси серверах, которые присутствуют на нашем хостинге. Данные с АПИ должны синхронизироваться с данными в БД. На хостинге уникальными значениями являются имена серверов.

Нужно создать несколько крон - задач, которые с периодичностью будут запускаться в веб приложении. Периодичность выполнения крон - задач:

- каждый час: веб приложение проверяет сроки завершения аренды прокси у юзеров, и если находит что у кого то из пользователей срока аренды завершился - отправляет запрос по АПИ на хостинг прокси для выключения данного прокси для пользователя и отвязку логина - пароля. Нужно учесть, что если у пользователя включен тариф “Безлимит на день”, то мы не отключаем ему доступ к прокси. Для этого нужно создать middleware. В общих случаях мы по сути отвязываем логин - пароль юзера от прокси и останавливаем докер - контейнер с данным прокси.
- каждый день: веб - приложение запускает генерацию новых прокси, используя команду от DevOps. Нужно предусмотреть возможность передачи в команду параметров для генерации (страна, порт, …).