В основу нашего проекта нам необходимо было создать уязвимость веб-сайта и сделать SQL инъекцию, чтобы получить доступ к базе данных.

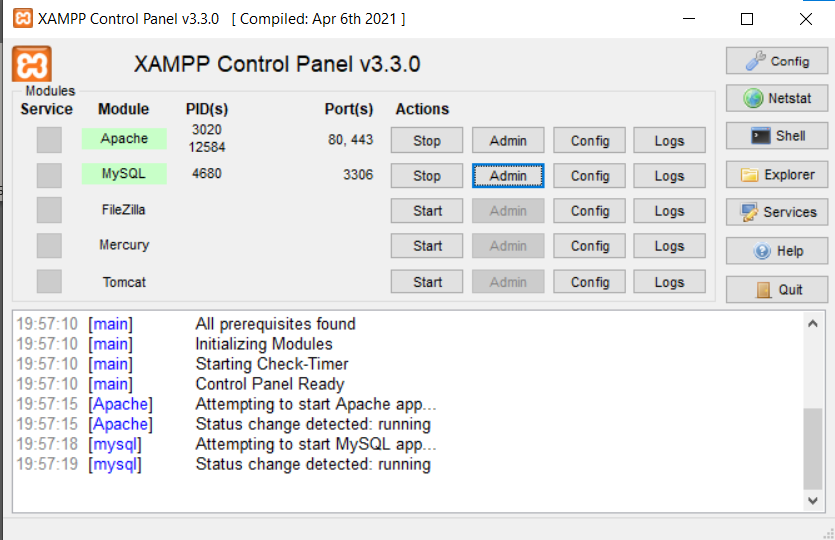
Мы сделали 2 разных версий сайта:

1) Первая версия — это сайт с уязвимостями для SQL инъекций, чтобы под видом злоумышленника авторизоваться на сайт и всячески изменять либо украсть данные из таблиц базы данных.

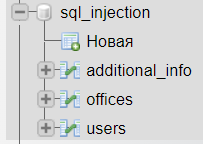
2) Вторая версия — это защищенный сайт, который полностью защищен от SQL инъекций, таким образом злоумышленник не подберется к нашему веб-сайту.

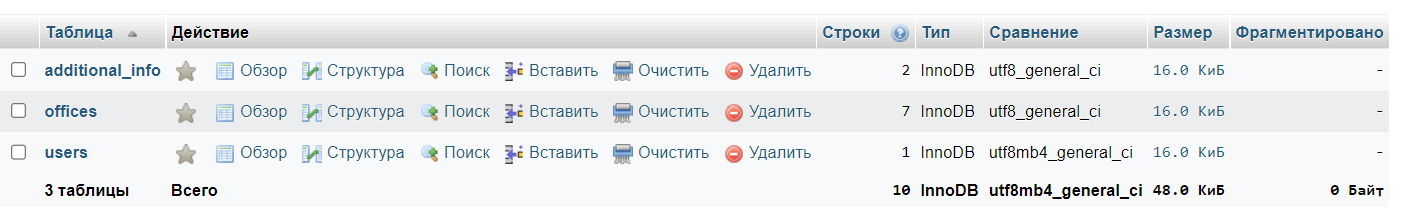
Наш веб-сайт написан на PHP и CSS, а используем базу данных MySQl.

Пользуемся XAMPP для локального сервера



Запускаем наши локальные сервера **Apache и MySQL**



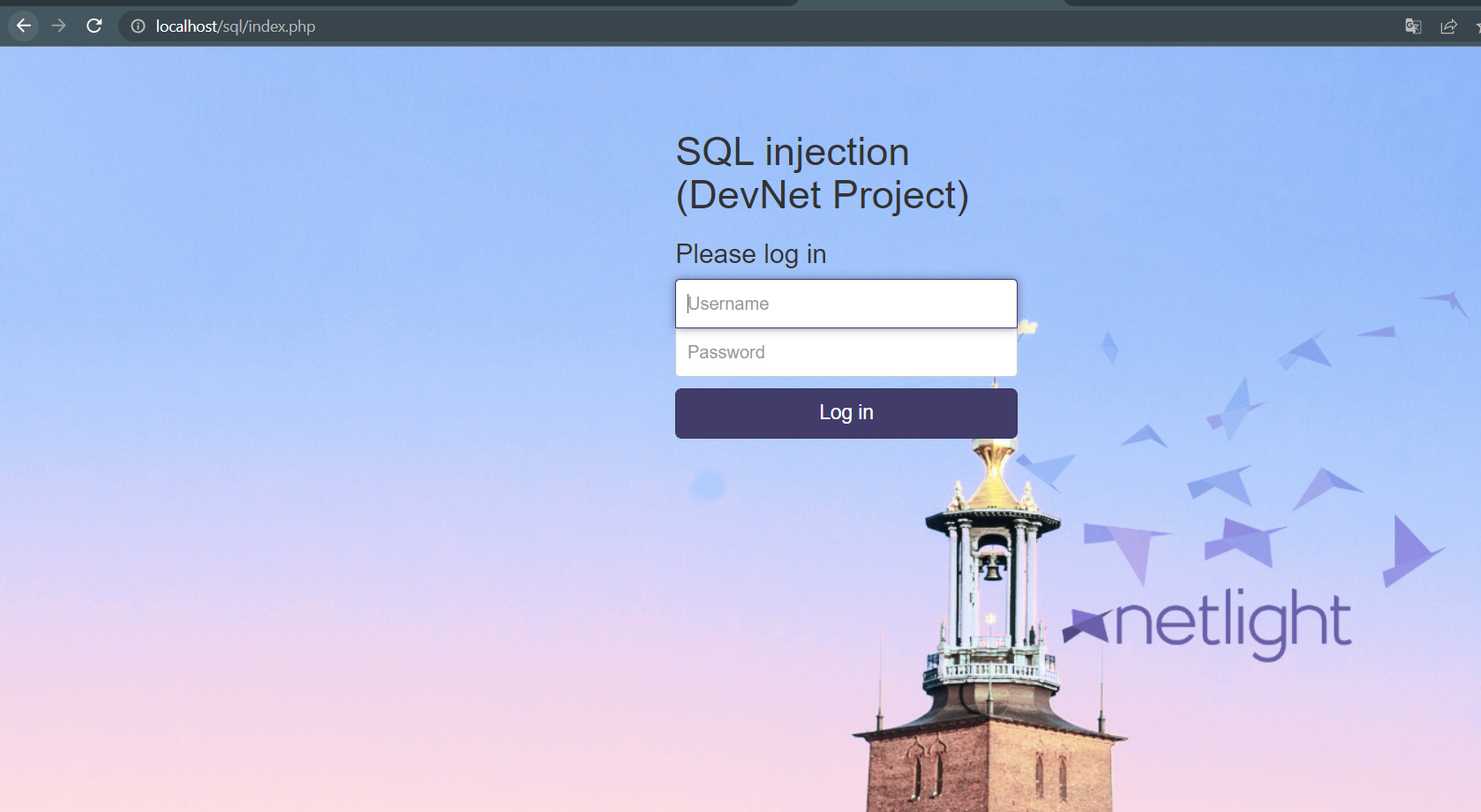


Наша база данных называется **sql\_injection** и имеет внутри 3 таблицы: additional\_info, offices & users.

Таблица users понадобится нам для авторизации, а additional\_info и offices для того, чтобы вытаскивать данные через поиск после авторизации.

Далее заходим на наш веб-сайт через localhost

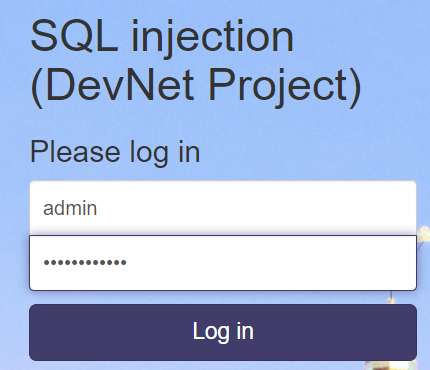
Попадаем на главную страничку для логина (авторизации) (прим. это у нас уязвимый для SQL инъекции сайт)



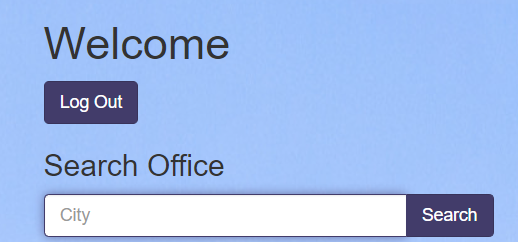
В поле username вводим admin

В поле с паролем вводим <password>' OR 1=1 -- </password> с пробелом в конце

(без <password> </password, добавил для наглядности, чтобы пробелы были видны)



Попадаем в страницу welcome.



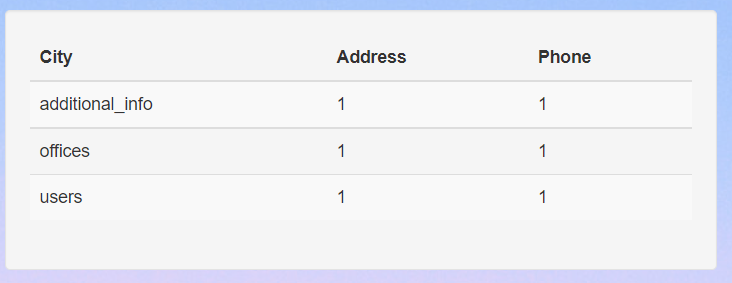
Далее в поле поиска вводим SQL запрос, чтобы вытащить данные с базы данных, на этот раз выводим все имеющиеся таблицы в базе данных sql\_injection.

Вводим следующий запрос:

<code>' UNION ALL SELECT table\_name, 1, 1 FROM information\_schema.tables WHERE table\_schema=database() -- </code> также с пробелом в конце

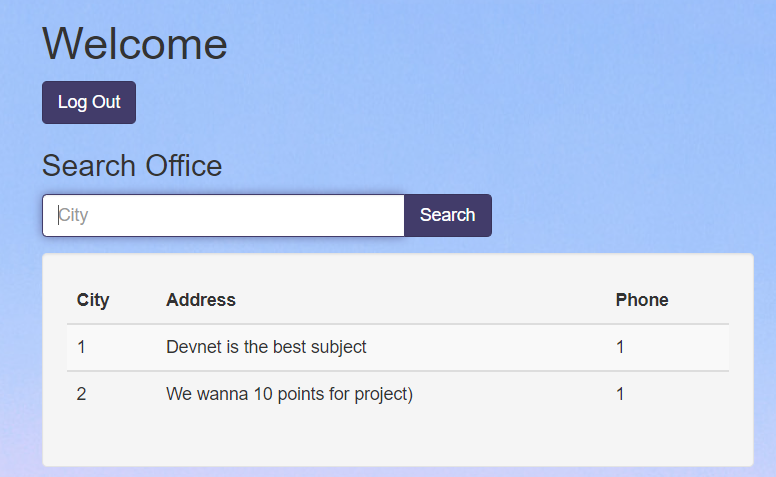
Получаем все таблицы, которые имеются в базе данных с именами: те же 3 таблицы,

additional\_info, offices, users.

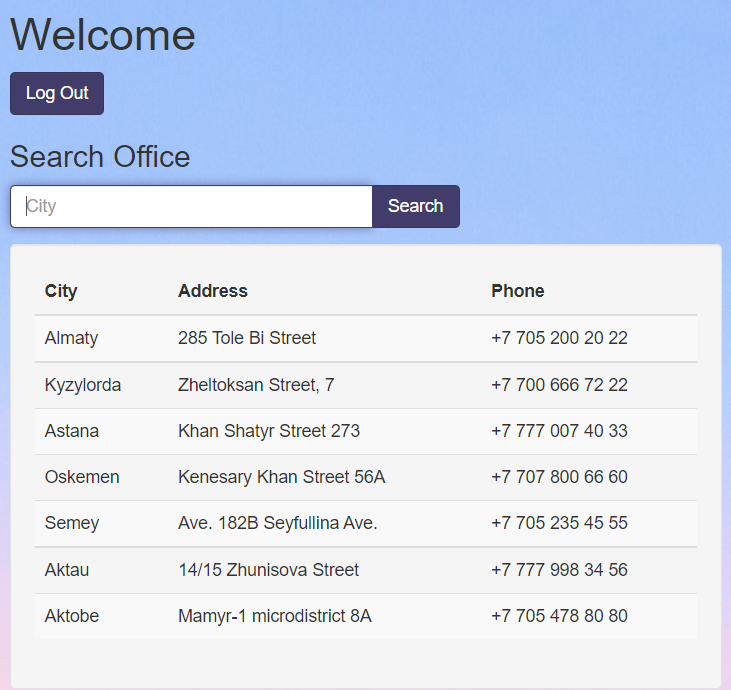


<code>' UNION ALL SELECT \*, 1 FROM additional\_info -- </code> также с пробелом в конце

Получаем данные с таблицы additional\_info.

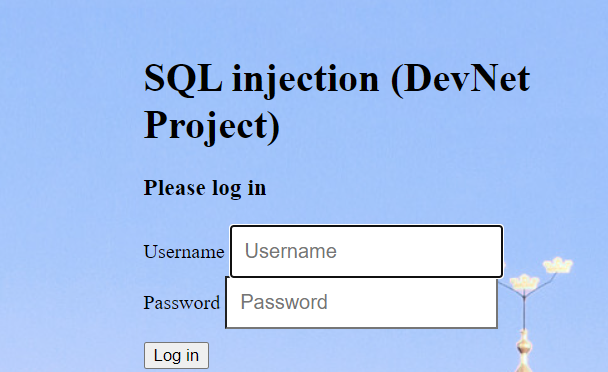


Также таким образом вытаскиваем данные с таблицы offices.



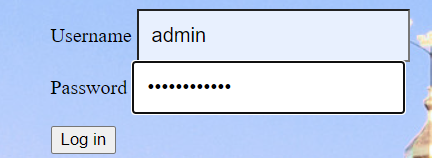
Теперь попробуем войти в защищенный от SQL инъекций сайт.



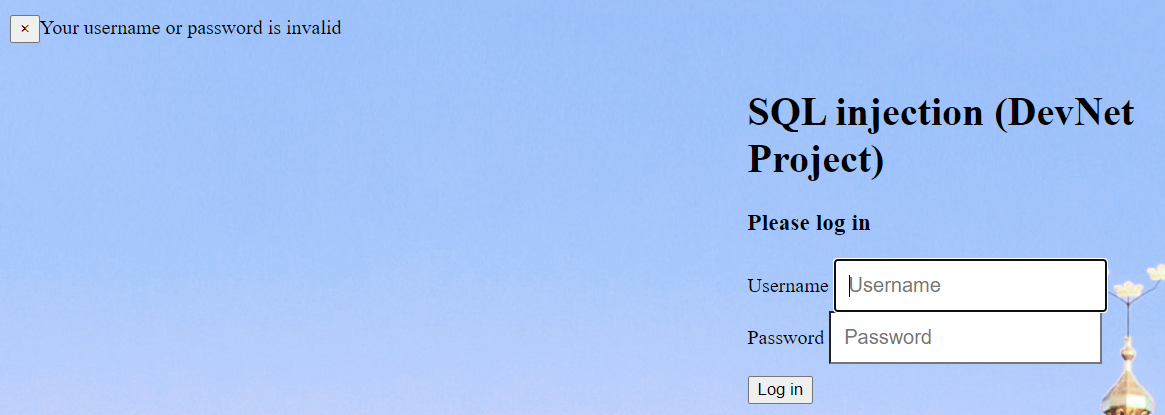


В поле username вводим снова admin

В поле с паролем опять попробуем ввести инъекцию и посмотрим, что получится. Вводим <password>' OR 1=1 -- </password> с пробелом в конце (без <password> </password, добавил для наглядности, чтобы пробелы были видны)



Логинимся и получаем отказ, делаем вывод, что наш веб-сайт защищен от SQL-инъекций.



Вот как выглядел наш запрос в уязвимой версии, тут были ординарные кавычки, которые уязвимы к SQL инъекциям и нет экранирования или любых других способов защиты от инъекций.

$query = "SELECT id FROM users WHERE password = '$password' AND username = '$username'";

$result = $db->query($query);

Далее мы используем один из лучших способов защиты от SQL инъекций -Подготовленные запросы. Суть в том, что SQL запрос сначала "подготавливается", а затем в него отдельно передаются данные.

Такой подход гарантирует отсутствие SQL-инъекций в момент подстановки данных, поскольку запрос уже "подготовлен" и не может быть изменён.

$stmt = $db->prepare("SELECT password FROM users WHERE username = ?");

$stmt->bind\_param('s', $username);

$stmt->execute();

$result = $stmt->get\_result();