**Our Project**

**Security of Website**

**Done by: Zhubatov Y.**

**Shazhabayev A.**

**Tursynbai A.**

**Anarbayev N.**

**Project Goals**

1. Get acquainted with SQL injection, introducing it into the website and come up with ways to protect against SQL injection, also do automation testing, deploying changes using CI/CD tools.
2. Create 2 versions of the website (first vulnerable to SQL injection, the second - SQL-protected)
3. Raise database and connect frontend to backend
4. To inject SQL injection into the website and perform an attack, manipulating the database in different ways (pulling, deleting, stealing, modifying)
5. Make a website secure from SQL injection

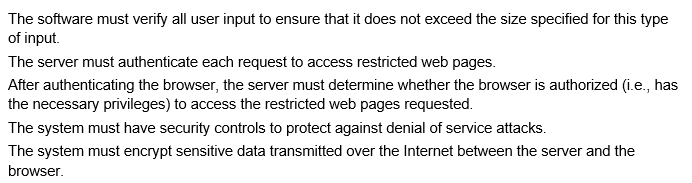
**Team and roles:**

|  |  |  |
| --- | --- | --- |
| **Team Member** | **Roles** | **Project Responsibilities** |
| **Yerdos Zhubatov** | **Project Manager** | Project documentation, project goals, project plan, timeline, budget, risk analysis, prepares reports, gives technical tasks to employees, and connects the client and project participants. |
| **Nurgissa Anarbayev** | **Project Tech Leader** | Creates two versions of the website (the first - vulnerable to SQL injection, the second - protected from SQL injection), develops site design, fixes bugs website prepares the site for testing. |
| **Abylai Shazhabayev** | **DevOps Engineer** | Engaged in the automation of our project, doing the automation of testing, deploying changes using CI/CD tools, namely Jenkins. Does PHPunit test all code snippets. Also, Jenkins test using SSH-key. Interfaces with AWS cloud, where Jenkins test with SSH-key checks our code for different changes every 10 minutes. |
| **Aibolat Tursynbai** | **QA Engineer** | Raises the database for the website, i.e., does the backend of the site. Adds data to the database, connects the frontend to the backend. Implements SQL injections into the website, writes SQL queries and SQL injections. |

Table

Description automatically generated

**Project Requirements**



Text, letter

Description automatically generated

## Strategy/Project Plan

Provide a brief description of your team’s strategy for completing this project.

Our plan is to complete the project by the end of November, going step by step, complying with all requirements. Everyone has been assigned roles and responsibilities, and everyone involved is doing a great job with their roles and tasks. All planned goals have been met and the project can be considered successfully completed.

We successfully built the website, implemented the SQL injection, tested in attacker mode, and also created a strong line of defense after that, to demonstrate a secure website. We also did various tests of our website by implementing automation in the project.

**Using GitHub for Collaboration**

What is the link to your GitHub repository?

<https://github.com/AbylaiShazhabayev/Security-of-Website>

Describe how GitHub was used to:

1. Describe/Create the master branch

GitHub is needed so that everyone involved in the project can upload their code there and collaborate in the development process. In other words, it is an environment for collaborative development in teams, each participant has its own branch that contains data (code) of each member of our project.

* The Documentation branch is the Project Manager branch of Yerdos
* DevOps branch - Abylai's DevOps branch.
* Frontend branch - Nurgisa's TechLead branch
* SQL branch - Aibolat's QA Engineer branch

1. Describe/Create your branch

<https://github.com/AbylaiShazhabayev/Security-of-Website/tree/Documentation>

In the documentation branch I keep all the necessary documentation for the project, I uploaded all the project parts, necessary reports and presentations there, since I am the project manager. There is a full description of each part of the project, what we did and how we did it.

Describe your team's application:

1. Which features of the whole project did you choose to implement from the backlog? Describe your personal task (code or document processing) and expected results of your part

**I chose the function of project manager; I am responsible for the maintenance of all project documentation. That is, all the steps taken by my colleagues will be recorded by me. As a result, we developed a website (or rather its 2 versions) 1-site vulnerable to SQL injection, 2-site fully protected from it, raised the database, did project testing, created for each team member a branch on GitHub, where everyone published his part and worked on it. And I, as the project manager, in turn wrote everything down and kept documentation of the project.**

1. What are the specific objectives of your features and how do you plan to test them? Describe the methods of testing and expected results of your tests

**The goal of our features is to automatically test our code. To test our code in Jenkins, we wrote Jenkins Pipeline and used bash scripts.**

**As a project manager, I do not need to test anything, but I write down all the documentation and record the testing process and other steps taken by team members and describe everything.**

1. Why were your inputs stored? Please apply the link to your branch on github.com or other system of version control

**My branch is called documentation and contains all of the project documentation.**

<https://github.com/AbylaiShazhabayev/Security-of-Website/tree/Documentation>

**Strategy of Automation Development of your application**

Provide a brief description of your team’s strategy for automation this project.

What CI/CD tool did you use for your project? Please attached the jobs description and expected results.

**We take our code from GitHub, then connect it to Jenkins using SSH-key. We then build our code and test it. After a successful attempt, it checks for changes to our code every 10 minutes, if Jenkins detects any changes, it does a deploy to the AWS cloud. As a result, our code is fully tested and interacts with the AWS cloud.**

**Final Deliverables**

**Presentation**

Provide the following evidence (which will also be part of your presentation):

Application and Git:

* Application code with new features
* Show changes pushed to GitHub
* Test cases
* CI/CD pipeline functionality

Reflect on the type of issues you have faced while working on this project. How did you find solutions and what have you learned?

**As a project manager I have to carefully describe all the processes and maintain documentation and I didn't have any particular problems, since I don't have to write code, I didn't have any errors either. There was just a lot of, shall we say, paperwork.**

**Что такое SQL-инъекции и как от них защититься?**

SQL-инъекциями называют атаки, позволяющие злоумышленнику производить различные несанкционированные действия над базой данных. Они могут затрагивать как сами данные, так и структуру базы. Для этого в качестве входных данных передаются специальные строки, содержащие вредоносные команды. Приложение, уязвимое к SQL-инъекциям, производит вставку этих строк в шаблон запроса без проведения необходимых проверок. В результате формируется запрос, выполняющий действия, определённые злоумышленником. При этом запрос будет являться корректным с точки зрения синтаксиса SQL.

SQL-инъекция может привести к следующим последствиям:

* Раскрытие конфиденциальных данных. Атакующие могут заполучить конфиденциальную информацию, хранящуюся на SQL-сервере.
* Компрометация целостности данных. Злоумышленники могут отредактировать или удалить информацию в вашей системе.
* Нарушение приватности пользователей. В зависимости от того, какие данные хранятся на SQL-сервере, атака может привести к раскрытию конфиденциальных пользовательских данных – адресов, номеров телефонов и сведений банковских карт.
* Получение злоумышленниками административного доступа к вашей системе. Если у пользователя базы данных есть привилегии администратора, с помощью вредоносного кода атакующий может заполучить доступ к системе.

**Как защититься от SQL-инъекций?**

* **Приведение к целочисленному типу**

В SQL-запросы часто подставляются целочисленные значения, полученные от пользователя. В примерах выше использовался идентификатор города, полученный из параметров запроса. Этот идентификатор можно принудительно привести к числу. Так мы исключим появление в нём опасных выражений. Если хакер передаст в этом параметре вместо числа SQL код, то результатом приведения будет ноль, и логика всего SQL-запроса не изменится.

* **Подготовленные выражения**

Вид атак типа «SQL-инъекция» возможен, потому что значения (данные) для SQL-запроса передаются вместе с самим запросом. Так как данные не отделены от SQL-кода, они могут влиять на логику всего выражения. К счастью, MySQL предлагает способ передачи данных отдельно от кода. Такой способ называется подготовленными запросами.

* **Не пользуйтесь методом GET в формах**

Передавать переменные этим способом очень опасно, потому что они оказываются на виду у пользователей. Если информация важная, то лучше используйте POST. Из ссылки злоумышленник может узнать не только имена переменных, но и какие значения должны быть в них — так он сможет выбрать оптимальную переменную для ввода инъекции.

### **Используйте PDO**

С помощью PDO и плейсхолдеров можно значительно снизить риск инъекции, потому что данные и запрос отправляются отдельно. Сначала производится подключение к базе, потом подготавливается запрос, затем отдельно указываются переменные, и, наконец, запрос выполняется

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

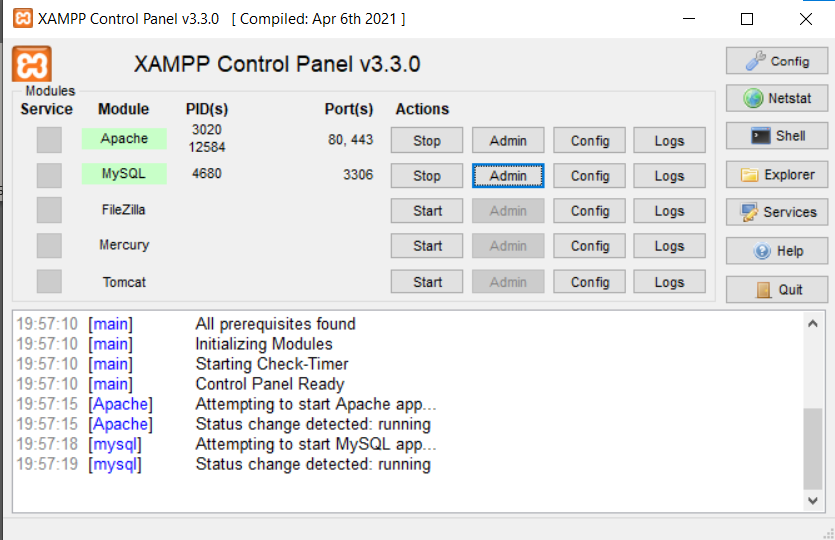
Мы сделали 2 разных версий сайта:

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.

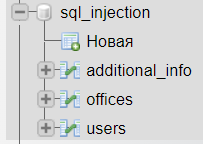
2) Вторая версия — это защищенный сайт, который полностью защищен от SQL инъекций, таким образом злоумышленник не подберется к нашему веб-сайту.

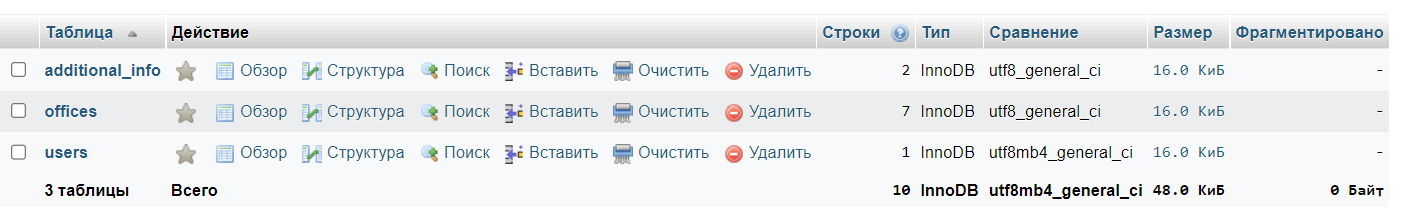
Наш веб-сайт написан на PHP и CSS, а используем базу данных MySQL.

Пользуемся XAMPP для локального сервера



Запускаем наши локальные сервера **Apache и MySQL**



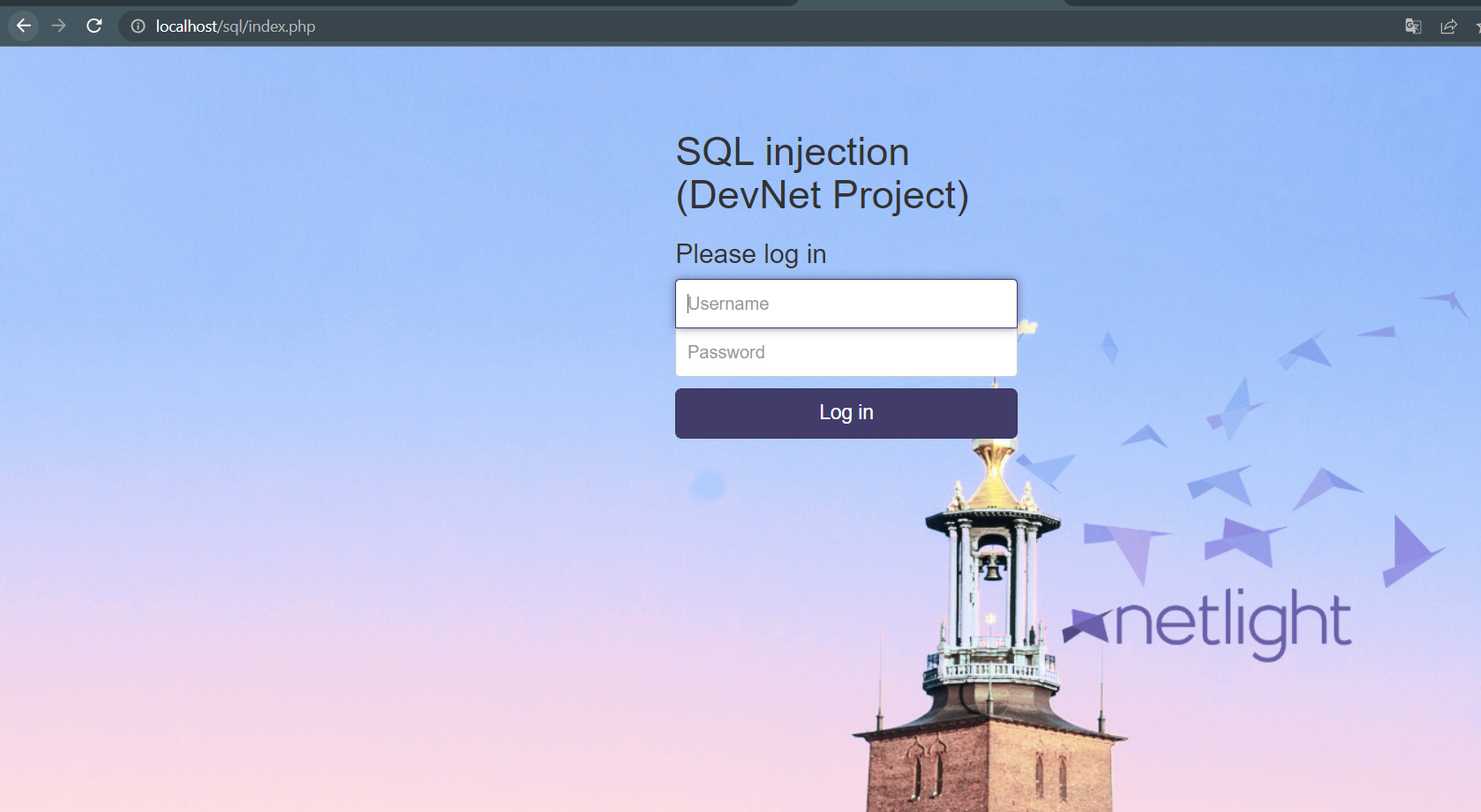


Наша база данных называется **sql\_injection** и имеет внутри 3 таблицы: additional\_info, offices & users.

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)



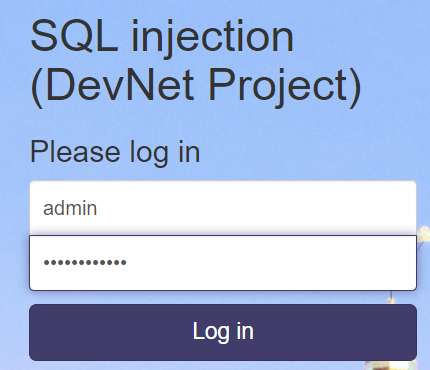
**В поле username вводим, к примеру admin, можем взять любой логин (yerdos или же), так как в итоге с помощью SQL-инъекции в окошке пароле нас в любом случае пропустят в систему, с результатом TRUE, так как мы прописали в инъекции “OR 1=1” что даст нам положительный результат.**

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

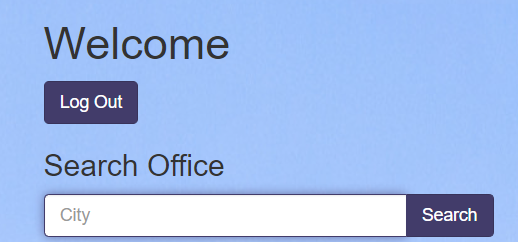
(без <password> </password, добавил для наглядности, чтобы пробелы были видны)

Изображение выглядит как стол

Автоматически созданное описание



Попадаем в страницу welcome.



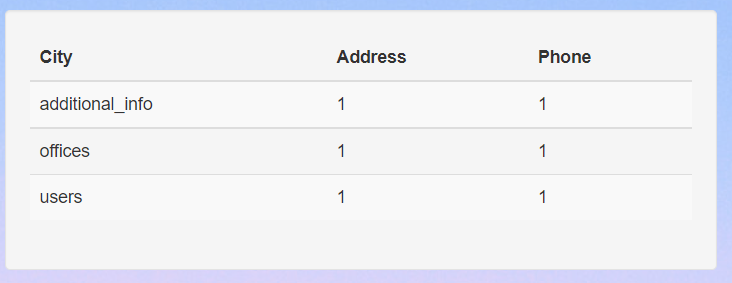
Далее в поле поиска вводим SQL запрос, чтобы вытащить данные с базы данных, на этот раз выводим все имеющиеся таблицы в базе данных sql\_injection.

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

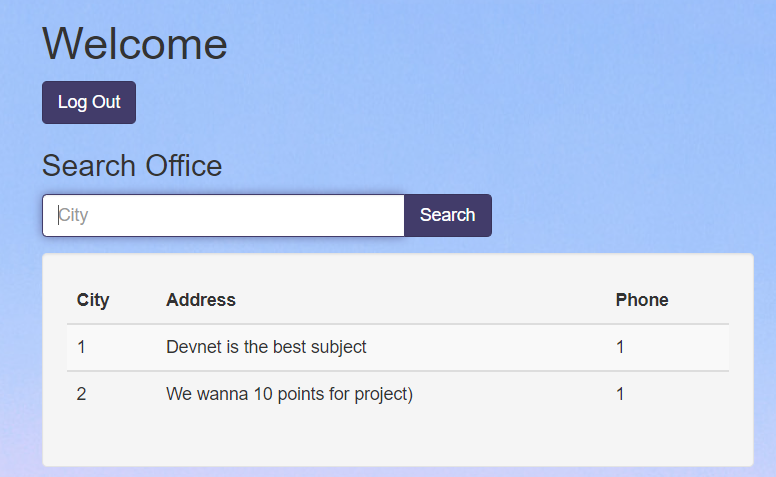
Получаем все таблицы, которые имеются в базе данных с именами: те же 3 таблицы,

additional\_info, offices, users.

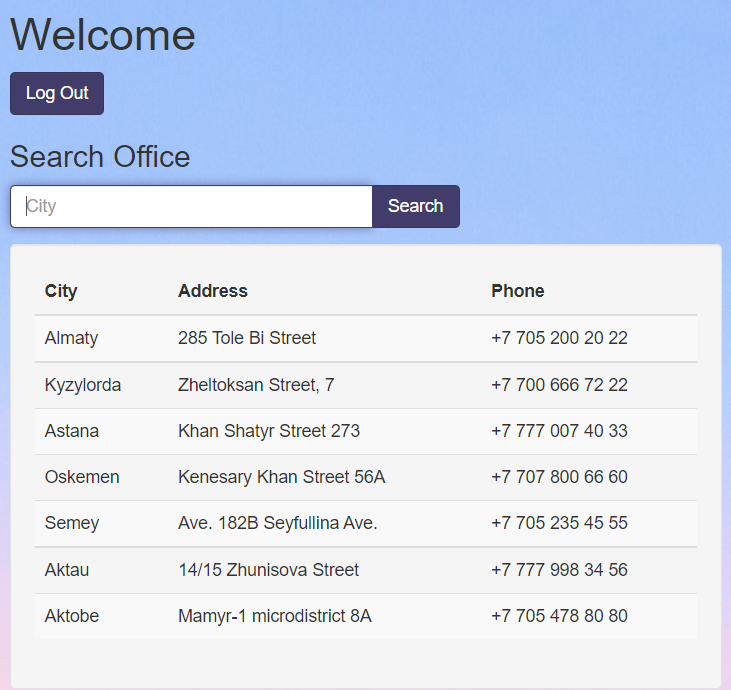


<code>' UNION ALL SELECT \*, 1 FROM additional\_info -- </code> также с пробелом в конце

Получаем данные с таблицы additional\_info.

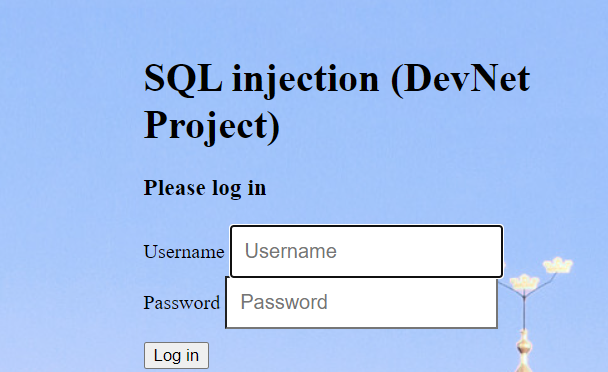


Также таким образом вытаскиваем данные с таблицы offices.



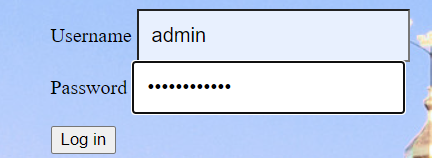
Теперь попробуем войти в защищенный от SQL инъекций сайт.



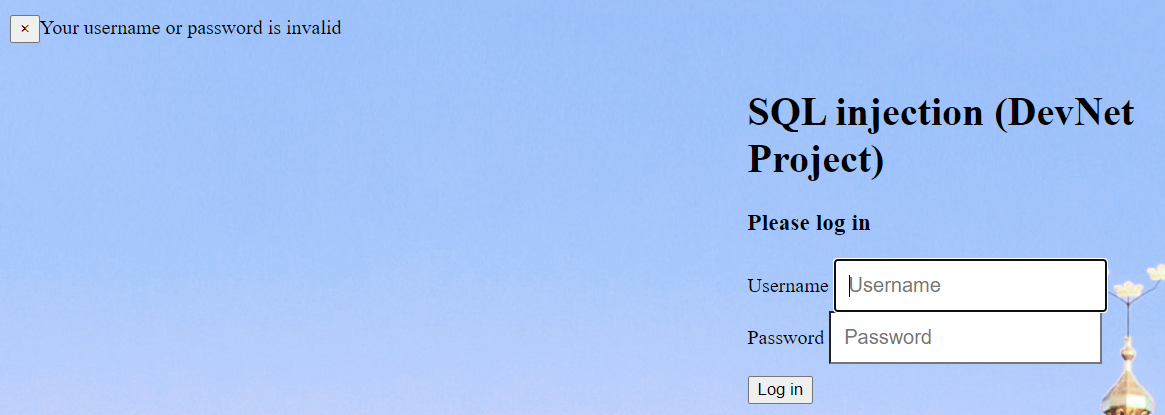


В поле username вводим снова admin

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();

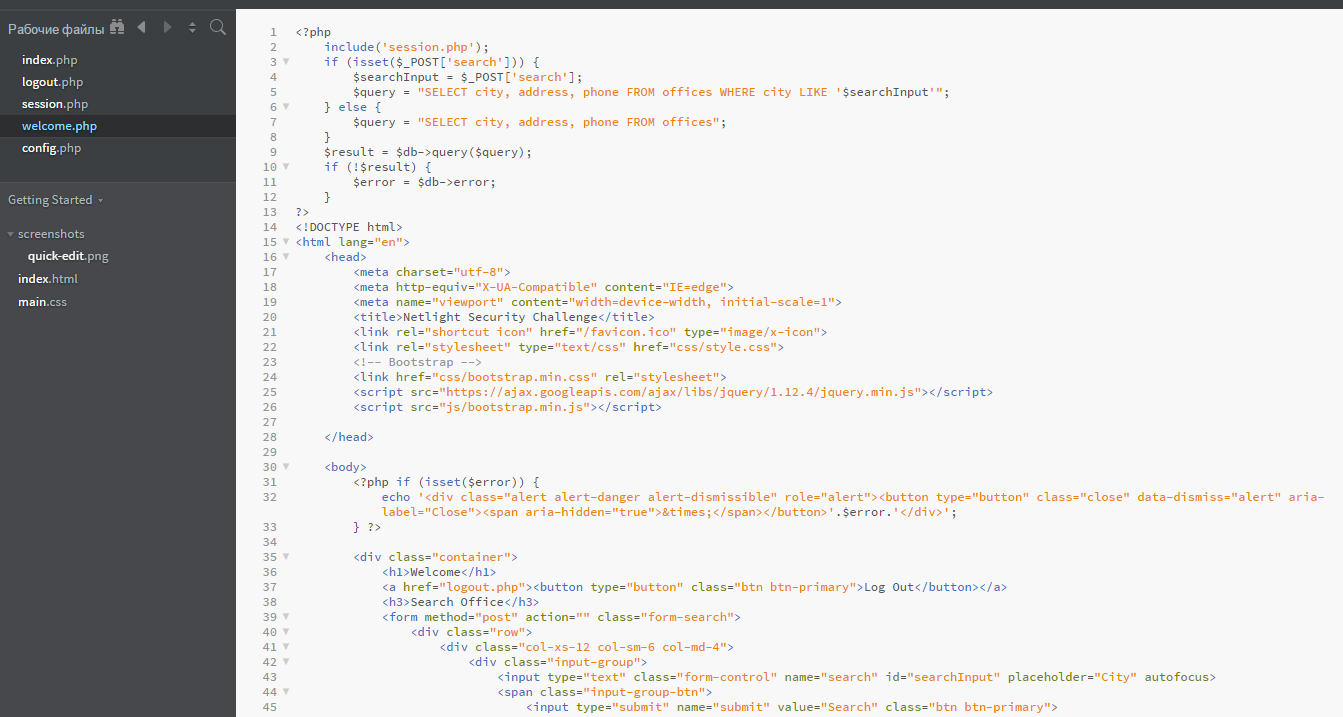
**Frontend code:**

**“index.php” directory, the main page of the website where the login form is stored.**

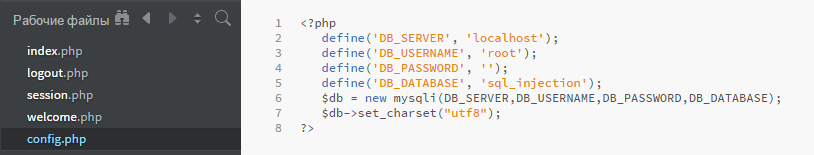




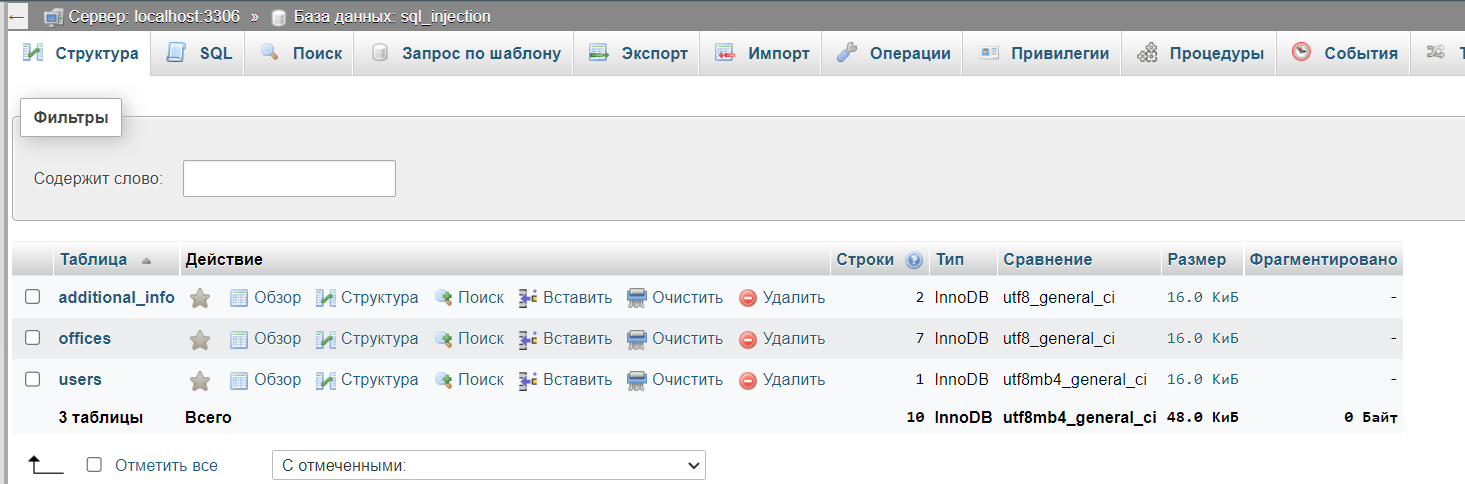
**“welcome.php” directory, the next page of the website, where you get after logging in, where there is also another form from which you can manipulate the database.**

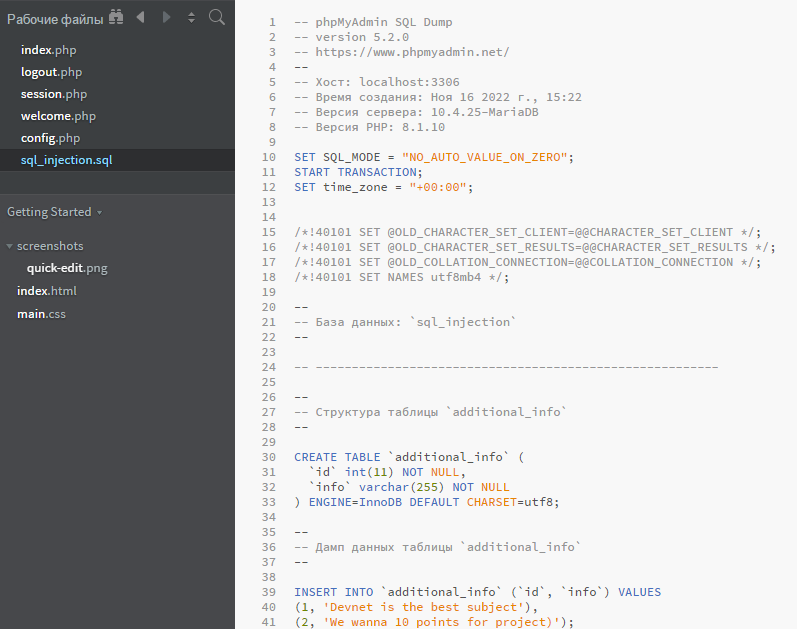


“config.php” directory, which connects the frontend to our database.



**Our database is called "sql\_injection" and some of its code snippets.**





# Project Activity 5 – Final Presentation of your App

Describe your team's application:

1. Which requirements did you and your team achieve in implementation?

We achieved all the necessary requirements and goals that were planned. We created the website, implemented SQL injection, did database manipulation, and successfully defended against it. We also implemented automated testing with Jenkins. Sent our code to AWS servers and got it up there.

1. What development and testing tools did you use?

We used the php, css programming language to create the frontend. We used Brackets to edit the code. Then to raise the database we used MySQL language, for the local server we used XAMPP software.

PHPunit a series of tests, running code snippets to automatically check the correctness of parts (modules) of the website.

We take our code from GitHub, then connect it to Jenkins using SSH-key. We then build our code and test it. After a successful attempt, it checks for changes to our code every 10 minutes, if Jenkins detects any changes, it does a deploy to the AWS cloud. As a result, our code is fully tested and interacts with the AWS cloud.

Record your algorithm, input and expected results

|  |  |
| --- | --- |
| **Input** | **Result** |
| **PHP and CSS code** | **Frontend of Website** |
| **MySQL database** | **Backend of Website** |
| **Connect Frontend and Backend** | **Finished Website (2 versions) vulnerable and secured** |
| **PHPunit test** | **Checked the website of all code fragments with PHPunit test.** |
| **Jenkins test using SSH-key** | **Passed through Jenkins testing for code changes.** |

**Strategy/Project Plan**

**Provide a brief description of your common team’s strategy for completing this project.**

Upon completion of the project, our chosen strategy worked well and we met all requirements and goals, our strategy was to meet deadlines and budgets by dividing responsibilities amongst ourselves.

## Final Deliverables

### Presentation

Provide the following evidence (which will also be part of your presentation):

* Application code
* CI/CD jobs used
* Example output of running code showing:
  + Input data before test
  + Expected result after test
  + Status in CI/CD tool

Reflect on the type of issues you have faced while working on this project. How did you find solutions and what have you learned?

**No special problems arose during the work, but even if they did, the project manager would have solved them quickly.**

**Team activities and reflection**

Deliverable/Rubric: Your manager is interested in knowing how everyone worked together as a team. Please do not forget to provide your evidence of team collaboration on GitHub or similar system version tool.

Below is a list of additional questions from your manager:

1. What did you enjoy about working as a team? What worked well?

**I liked the fact that everyone on the team shared their commitments and meticulously carried them out. When you divide a big task into smaller parts, it's easier to do it. It was the same with us, we divided the responsibilities between us, and everyone did what was required of him, and the result was a weighty project.**

1. What team problems did you encounter and how did you resolve them?

**No problems arose**

1. What technical problems did you encounter and how did you resolve them?

**There was a problem with XAMPP, where the MySQL server crashed, solved by replacing the wrong files with new ones by creating a new directory in the local MySQL directory.**

**Amazon banned our account because we had created many servers on their platform and had to pay 10 dollars to unbanned them.**

1. How was each team member held accountable individually and for the team as a whole?

**We communicated through teams, where everyone demonstrated their part of the project, and through GitHub we interacted with our branches.**

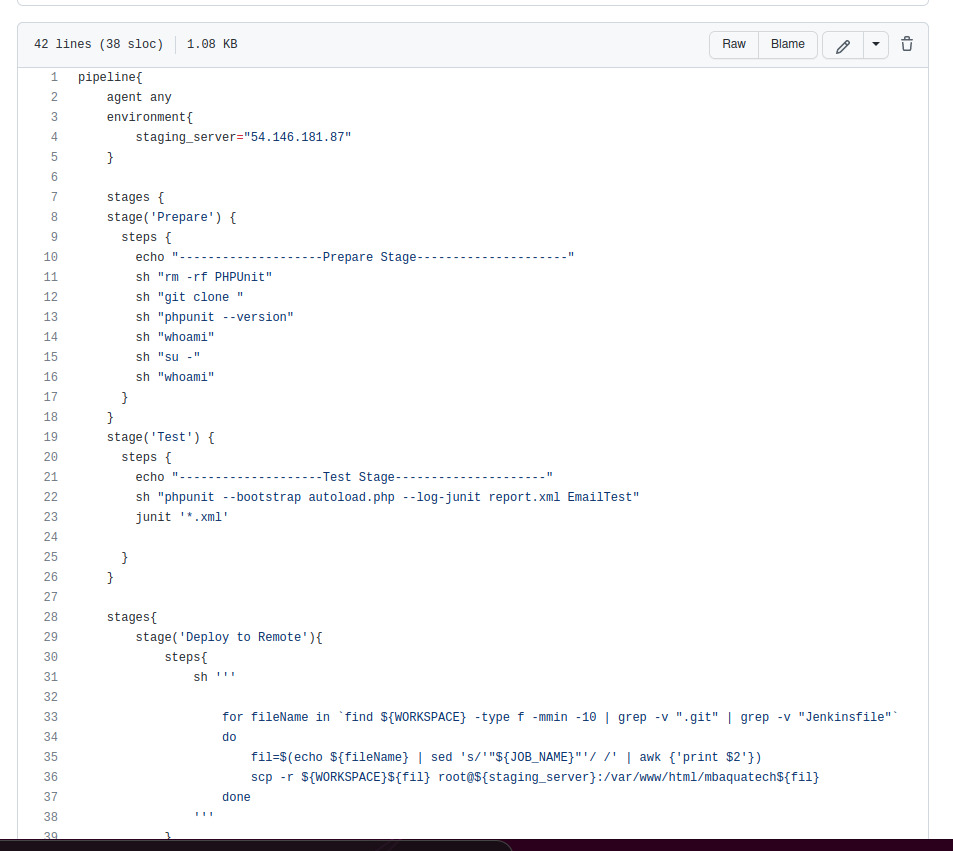
1. What was your team's decision-making process?

**The project manager Yerdos was responsible for the decision-making process, there were also general meetings to clarify situations.**

1. Overall, how were the team dynamics and what were any lessons learned?

**The dynamics of the team was intense, the tasks were not easy, and they were solved, we learned the lessons that we need more interaction as a team.**

Это наш код pipeline, который делает PHPunit тест, он проверяет каждые 10 минут на наличие изменений, и далее автоматически делает deploy на наш AWS сервер.



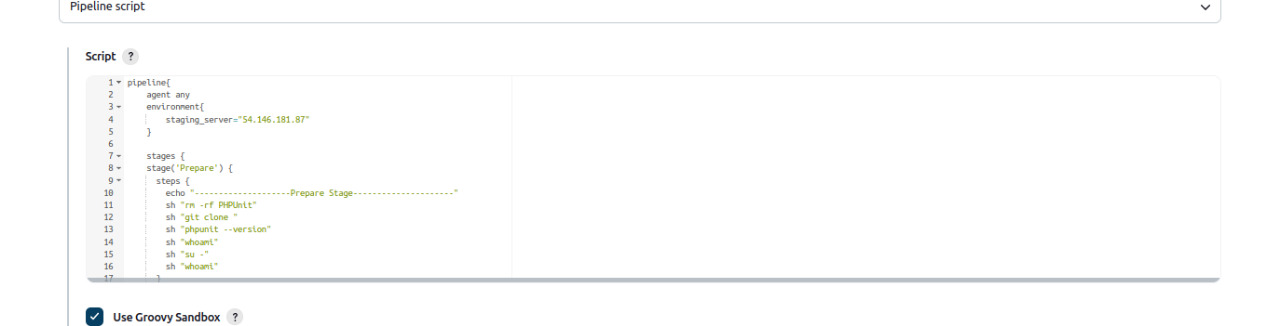
К примеру, мы тут сделали PHPunit тест на “welcome.php”, чтобы наглядно продемонстрировать логику тестирования.



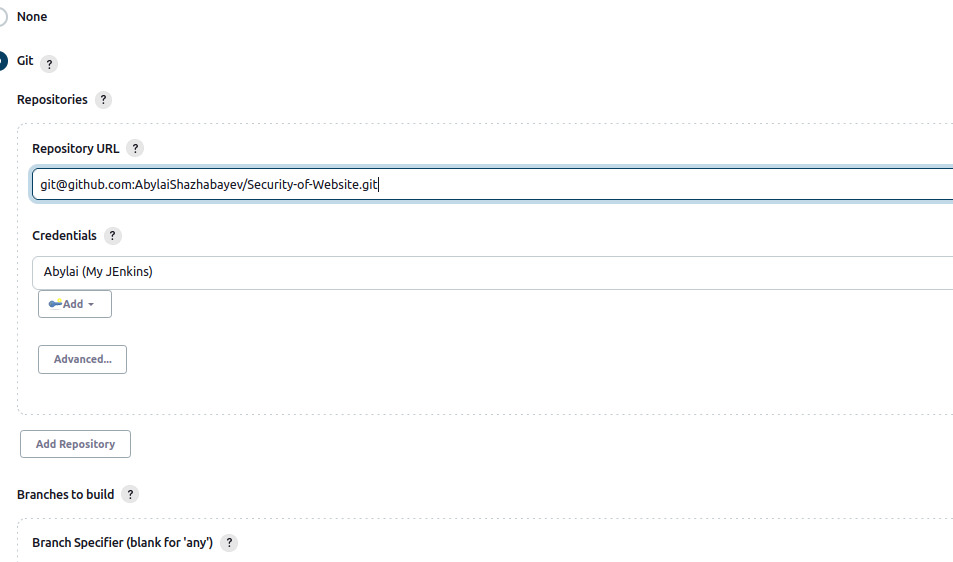
Скрипт pipeline с Jenkins, который делает deploy на удаленный сервер AWS.



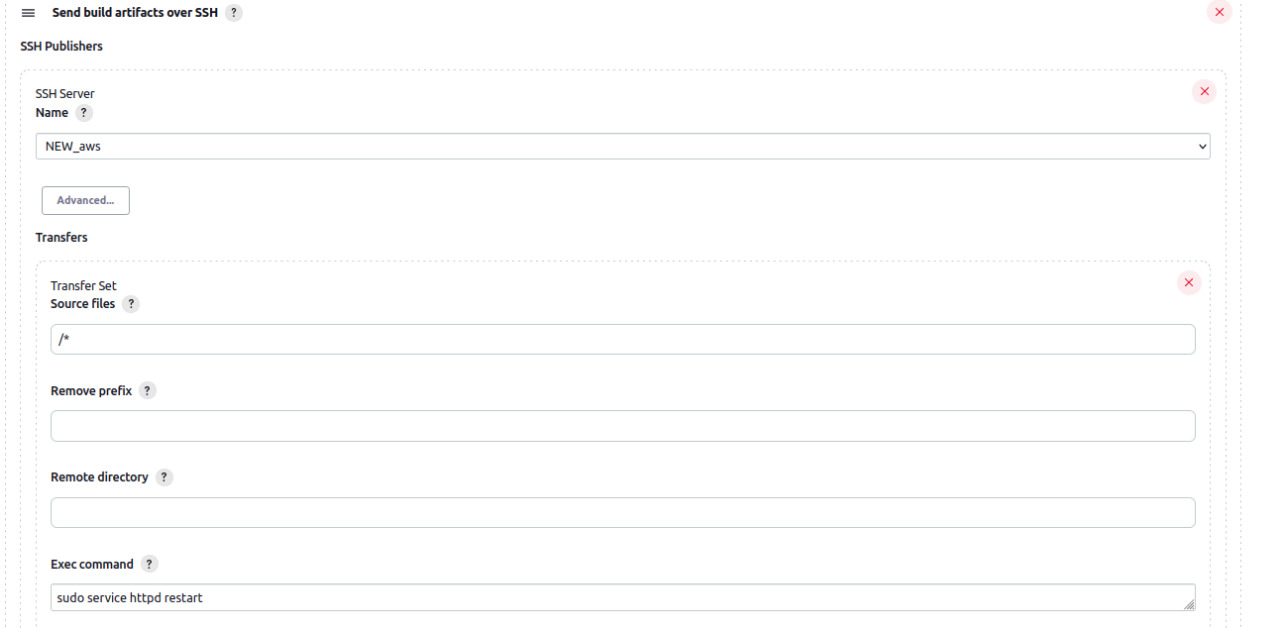
Как мы видим, наш скрипт pipeline, разделяется на stages, чтобы провести тестирование. И также указывается наш сервер AWS, его IP-адрес: 54.146.181.87



Тут показано как мы соединили GitHub и наш Jenkins, с помощью которого мы и делали наши тесты.



Здесь мы с помощью SSH-key соединили Jenkins с сервером AWS.



Это наша картина итоговых результатов проведенных тестирований, как мы можем видеть по build history, все последние тестирования горят зелеными индикаторами, что в свою очередь показывает, что все тестирования были успешны

