



**Реферат**

Отчет по производственной проектно-технологической практике: 21 страница, 16 рисунков, 4 используемых источника.

хеширование, пароли, «соль», криптографические методы, sha1, контейнеризация, документация, swagger, docker, dockerfile, http-запросы, запрос get, API, контроллер, mvc, visual studio

Объектом является хеширование пароля и соли для сохранности данных.

Целью работы является готовый сервис для генерации хеша на основе пароля и соли по заданным параметрам.

Основные полученные результаты:

* алгоритм генерации пароля и соли по заданным параметрам;
* получение хеша на основе алгоритма SHA1;
* обработка ошибок в следствии неправильно заданных параметров;
* возможность доступа к сервису через API.

**Содержание**

[Введение 8](#_Toc112789921)

[1 Задание на производственную проектно-технологическую практику 9](#_Toc112789922)

[2 Выполнение задания 10](#_Toc112789923)

[2.1 Написание основного кода программы 10](#_Toc112789924)

[2.2 Добавление сервиса в контейнер 13](#_Toc112789925)

[2.3 Документация 15](#_Toc112789926)

[3 Тестирование 17](#_Toc112789927)

[Заключение 20](#_Toc112789928)

[Список литературы 21](#_Toc112789929)

# Введение

В ходе прохождения производственной проектно-технологической практики была поставлена цель, а именно готовый сервис для генерации хеша на основе пароля и соли по заданным параметрам.

В первой главе рассказывается о задании, которые было выдано в ходе проектно-технологической практики.

Вторая глава содержит информацию о выполнении поставленной задачи: API, документации, контейнеризации и т.д.

Третья глава рассказывает о тестировании созданного в ходе выполнения задания сервисе, о принципах его работы.

# 1 Задание на выполнение

В ходе производственной проектно-технологической практики необходимо спроектировать сервис генерации числового пароля. Ниже представлены пункты, которые необходимо учесть в процессе проектирования:

1. Через API получать числовой код длинной 4-8 символов (кратно 2-м), «соль» и хеш (SHA1) от этих данных.

2. Длина пароля и «соли» должна передаваться через параметры запроса.

3. Запрос и ответ в формате JSON.

Дополнительные пункты задания:

1. Встроенные метрики Prometheus для мониторинга сервиса (кол-во запросов, коды ответов и время выполнения запросов);

2. Сервис необходимо сложить в контейнер;

3. Документация (ReDoc, Swagger).

# 2 Выполнение задания

## 2.1 Написание основного кода программы

Для создания сервиса было выбрано приложение Веб-API ASP .NET Core в среде программирования Visual Studio. В основе работы приложения лежит модель MVC. Для работы модели основные действия работы программы должны находиться в контроллере, в том числе метод контроллера должен содержать определенные атрибуты, которые позволят обращаться к сервису через публичный API. Запрос будет производиться в формате http, с помощью метода GET.

MVC (модель-представление-контроллер) – это способ организации кода, который предполагает выделение блоков, отвечающих за решение разных задач. Один блок отвечает за данные приложения, другой отвечает за внешний вид, а третий контролирует работу приложения.

Можно выделить следующие компоненты модели MVC:

Model (модель) – компонент отвечает за данные и определяет структуру приложения.

View (представление) – компонент отвечает за взаимодействие с пользователем: определяет внешний вид html-страницы и способы ее использования.

Controller (контроллер) – компонент отвечает за связь между моделью и представлением. Код компонента контроллер определяет каким образом сервис будет реагировать на действия пользователя [1].

Для взаимодействия с сервисом используются http-запросы. Самые распространенные методы GET и POST, которые позволяют получить или отправить данные с сервера. Для каждого случая используется свой метод, так как один и тот же метод в разных случаях может оказаться абсолютно небезопасным и неудобным.

GET используется для получения данных с сервера. Он передает серверу, что хочет получить клиент, какой документ прочитать. Именно на такой запрос и будет отвечать сервер.

В том числе, при использовании запрос GET есть возможность указать параметры запроса. Например, при использовании сервиса должны быть переданы данные о длине пароля и «соли». При передаче запроса без параметров, параметры устанавливаются по умолчанию.

Для передачи ответа пользователю в формате JSON, необходимо создать класс, объект которого будет содержать все необходимые данные и передавать их в заданном формате. Такой класс будет содержать данные о пароле, «соле» и хеше (см. рис. 1).

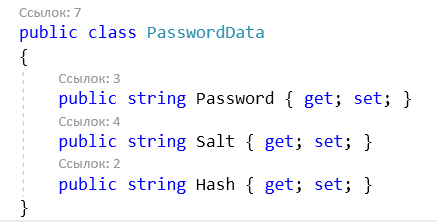


Рисунок 1 – Класс для хранения данных

Принцип работы основного алгоритма программы:

1. Корректировка длины пароля (если пользователь передал данные с ошибкой, либо не передал вовсе). По заданию пароль должен соответствовать следующим требованиям: длина пароля от 4 до 8 символов, длина кратна двум.

2. Корректировка длины соли – аналогично предыдущему пункту длина обрабатывается при неверных входных данных, либо при их отсутствии. В случае, если пользователь не передал данные о длине «соли», по умолчанию ее длина устанавливается равной 1, так как «соль» не может являться пустой строкой.

3. Задание случайного пароля заданной и уже скорректированной длины на основе функции генерации рандомных чисел.

4. Задание «соли» заданной длины, на основе методов криптографической библиотеки.

5. Создание хеша на основе заданных пароля и «соли». Для хеширования применяется функция SHA1.

Sha1 (или Secure Hash Algorithm) – это криптографическая хэш-функция, которая принимает входные данные и создает 160-битное (20-байтовое) хэш-значение, известное как дайджест сообщения, обычно отображаемое как шестнадцатеричное число длиной 40 цифр. Он был разработан агентством национальной безопасности США и Федеральному стандарту обработки информации США. С 2005 года SHA-1 не считается защищенным.

Для того, чтобы сгенерировать пароль, «соль» и хеш, необходимо обратиться к API сервиса, используя метод http-запроса - GET.

HTTP запрос состоит из трех основных частей, которые идут в порядке, приведенном ниже. Между заголовками и телом сообщения находится пустая строка (в качестве разделителя), она представляет собой символ перевода строки.

1. Строка запроса – указывается метод передачи, URL-адрес, к которому необходимо обратиться и версию протокола HTTP.

2. Заголовки – описывают тело сообщений, передают различные параметры и другие сведения, и информацию.

3. Тело сообщения – это сами данные, которые передаются в запросе. Этот параметр является необязательным и может отсутствовать [2].

При обращении к API сервиса без параметров запроса (см. рис. 3), длина пароля и «соли» устанавливается по умолчанию, и пользователь получить следующие результаты запроса:

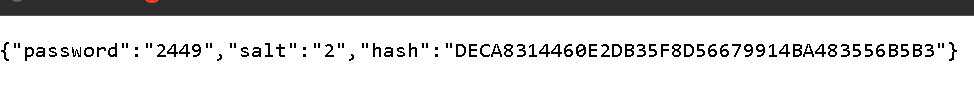


Рисунок 2 – Результат обращения к API

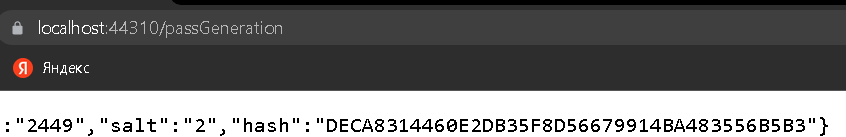


Рисунок 3 – Запрос к API без параметров

При обращении к API с заданными параметрами (см. рис. 5), пользователь получит следующий результат:



Рисунок 4 – Результат запроса GET

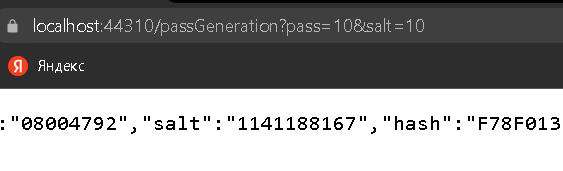


Рисунок 5 – Запрос GET

## 2.2 Добавление сервиса в контейнер

Контейнеризация — это более легкая форма [виртуализации](https://www.tadviser.ru/index.php/%D0%92%D0%B8%D1%80%D1%82%D1%83%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F) на уровне [операционной системы](https://www.tadviser.ru/index.php/%D0%9E%D0%A1), позволяющая запускать определённое приложение и системные библиотеки в разграниченной области. Эта область называется контейнер. Он содержит все компоненты, необходимые для запуска приложения, не зависимо от архитектуры серверной системы и работает с ОС используя стандартные интерфейсы.

Благодаря этой форме программы работают автономно с различными операционными системами, аппаратным обеспечением, базами данных и другими объектами. Эти приложения легко открываются в различных средах разработки и тестирования. Контейнер использует минимальные вычислительные ресурсы.

Одной из самых известных программных платформ с поддержкой контейнеризации является Docker. Контейнеры этой программы легко развертываются в облаке, где разработчики могут создавать любые среды разработки.

Этапы занесения сервиса в контейнер очень простые, а именно: для добавления, необходимо в первую очередь скачать необходимое приложение, установить и приступить к созданию образа, далее образ преобразуется в контейнер.

Для работы с Docker программа должна иметь файл под названием Dockerfile, который представляет собой некую инструкцию по сборке образа, содержащий команды: программы, зависимости и образы, необходимые для разворачивания образа[3].

В Visual Studio dockerfile образуется автоматически и для его создания не нужно создавать отдельный файл и писать туда команды для выполнения. Этот файл представлен на рисунке ниже:

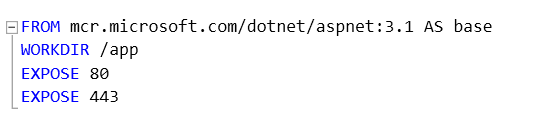


Рисунок 6 – Dockerfile

После создается образ, который затем запускается (рис. 7) на основе настроек (рис. 8).

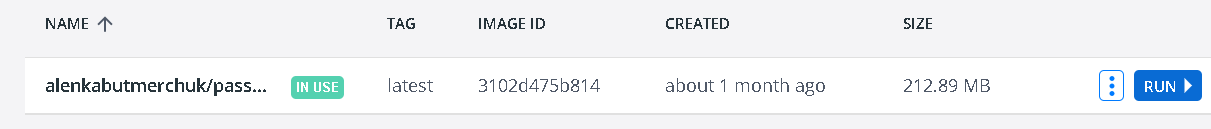


Рисунок 7 – Созданный образ сервиса

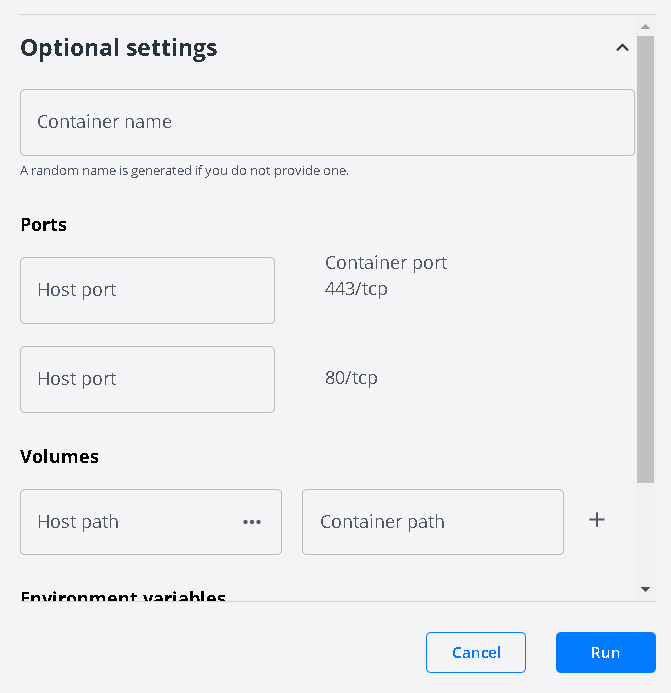


Рисунок 8 – Настройки образа

Далее на основе образа создается контейнер который и составляет основу контейнеризации. После всех выполненных операций появляется возможность открывать контейнер на разных операционных системах.

## 2.3 Документация

Для документирования сервиса был использован набор инструментов, который помогает описывать API, а именно Swagger..

Благодаря ему пользователи и машины лучше понимают возможности [REST API](https://highload.today/rest-api-soap/) без доступа к коду. С помощью Swagger можно быстро создать документацию и отправить ее другим разработчикам или клиентам которые будут использовать сервис.

В документации к сервису был написан следующий текст который представлен ниже, эта информация была записана в файл: название продукта: Сервис генерации числового пароля ProjectPass2. Сервис выполняет функции для генерации числового пароля, хеша и соли по входным данным. Предназначен для улучшения более крупного проекта или для использования в качестве отдельной модели. Сервис написан на языке C#, целевая платформа .NET 6.0., для запуска необходимо скачать visual studio 2022 после запустить проект через файл exe или sln.

Так же представлена документация для пользователя: сервис через API получает числовой код длинной 4-8 символов(кратно 2-м), соль и хеш. Длинна пароля и хеш передается через параметры запроса. Для получения данных необходимо запустить проект и ввести https://localhost:44342/Generic?ls=4&lp=6 где после слова Generic вводятся данные ls- это длинна соли, а lp - длинна пароля. После этого выводится наш сгененрированный пароль хеш и соль, которые можно использовать в других проектах. Запрос и ответ передавется в формате JSON. Для реализации документации использован Swagger. Для его отображения необходимо выполнить команду в браузере https://localhost:44342/swagger после чего появится окно с комментариями. В самой программе реализованны некоторые классы: class Optimal - отвечает за возврат и установку новых значений пароля,соли и хеша; class ClassPHS - отвечает за генерацию пароля, соли и хеша, а так же выполняющий проверку правильности введенных данных и их корректировку; так же есть класс Контроллера необходимый для передачи значений ls и lp.

# 3 Тестирование

В процессе тестирования необходимо выполнить несколько основных пунктов заданий, которые относятся к работе сервиса, а именно: является ли работа http-запроса корректной; полученные параметры от пользователя обрабатываются ли для корректной работы сервиса; при отсутствии передачи параметров сервиса, использует ли он параметры по умолчанию.

1. Выполнение проверки на корректность работы http-запроса:

Одним из способов, как можно отправить запрос по протоколу HTTP к серверу, является запрос методом GET.

Для создания запроса методом GET – необходимо вбить команду URL-адреса в адресную строку браузера (рис. 9).

Указывается локальный сервер и название контроллера содержащий метод с атрибутом.

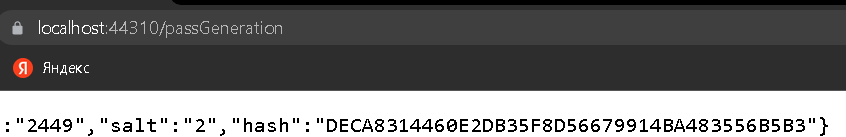


Рисунок 9 – Запрос GET

После выполнения запроса браузер получает следующий результат (формат JSON):

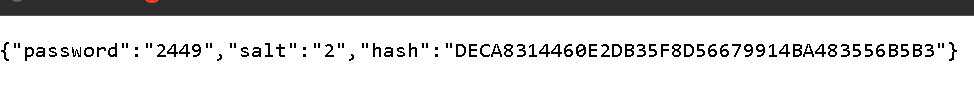


Рисунок 10 – Результат работы запроса

Работа запроса GET корректна, можно преступать к тестированию следующих аспектов сервиса[4].

1. Обработка параметров, которые не соответствуют условиям.

Для проверки обработки длины пароля передадим следующие длины пароля в запрос:

* длина пароля меньше нуля. Итоговая длина сформированная скорректированная сервисом: 4 (рис. 11);
* длина пароля меньше 4. Итог: длина 4 (рис. 12);
* длина пароля больше 8. Итог: длина 8 (рис. 13);
* длина пароля между 4 и 8, но не кратна двум. Итог: длина доводится до ближайшего, большего числа, кратного 2 (рис. 14).

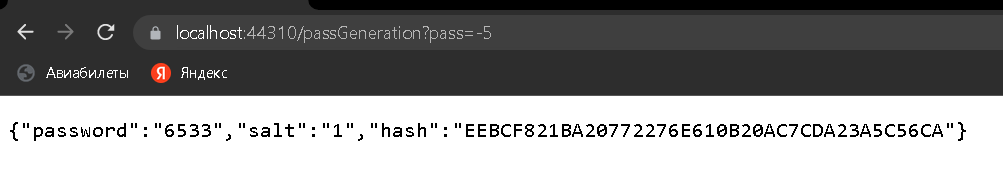


Рисунок 11 – Длина пароля -5

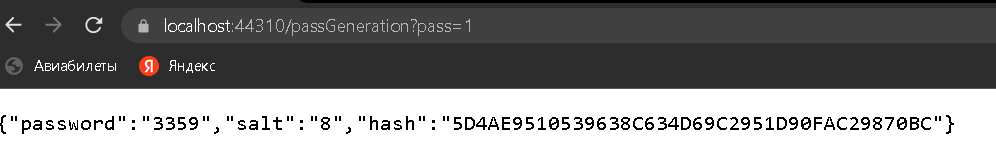


Рисунок 12 – Длина пароля 1

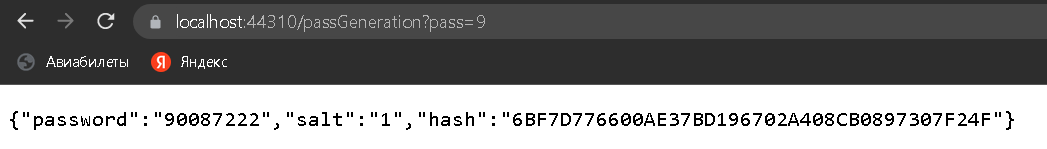
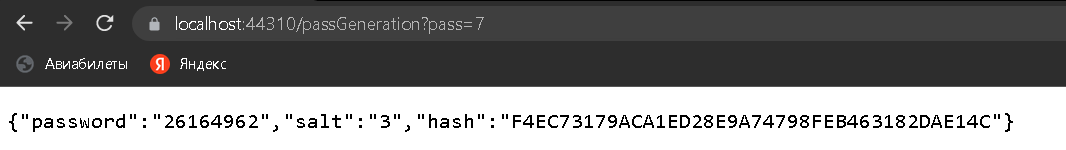


Рисунок 13 – Длина пароля 9

 Рисунок 14 – Длина пароля 7

Проверка длинны «соли». Указывается в параметрах число меньше нуля, либо равное нулю, длина должна равна 1 (рис. 15).

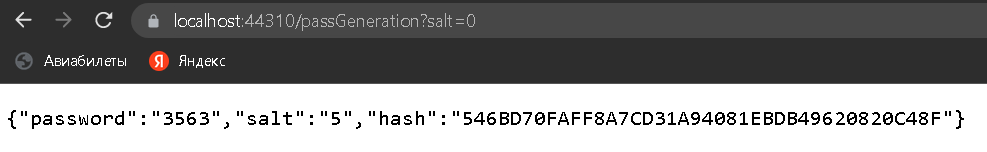


Рисунок 15 – Длина «соли»

Все параметры, переданные сервису с ошибками, были верно обработаны сервисом.

1. Отправка запроса без параметров, длины пароля и «соли» устанавливаются автоматически (рис. 16).

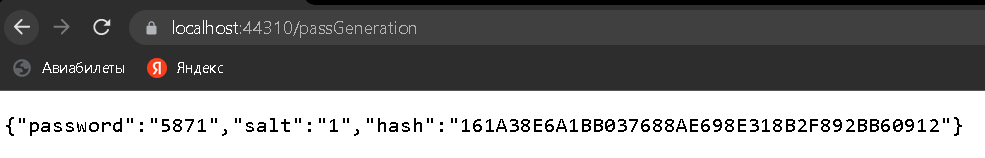


Рисунок 16 – Длины пароля и «соли» по умолчанию

После выполнения работы и отправки запроса без параметров в сервис длинна пароля и соли были установлены автоматически, а значит данная часть тестирования прошла успешно.

# Заключение

В результате прохождения производственной проектно-технологической практики получены следующие результаты:

* выполнена генерация пароля с помощью программы;
* выполнена генерация соли при помощи криптографических алгоритмов;
* выполнено создание хеша (алгоритм SHA1);
* выполнено создание методов корректировки длин параметров для правильной работы сервиса.

Полученные результаты можно использовать для достижения генерации пароля различной длины, а также для получения хеша от пароля и соли. Эти результаты могут быть использованы любыми приложениями и предприятиями для которых важна качественная генерация и использование в своих приложениях сервиса для генерации.

# Список литературы

1. Дженнингс, Ф. Практическая передача данных. Методы, сети и протоколы / Ф. Дженнингс. - М.: Мир, 1989. - 272 c

2. Фримен Адам.ASP.NET Core MVC 2 с примерами на C# для профессионалов, 2019. – 1008 с.

3. Моуэт Эдриен. Использование Docker, 2017. – 354 с.

4. Рекс Блек. Ключевые процессы тестирования - М.: Издательство Лори, 2014. - 544 с.