Министерство образования Республики Беларусь

Учреждение образования  
 БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
 ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерного проектирования

Кафедра проектирования информационно-компьютерных систем

Анализ и оптимизация бизнес-процессов

Отчет

по результатам выполнения задания к практическому занятию №4

на тему:

## 

**МЕТОДЫ РЕФАКТОРИНГА КОДА**

|  |  |  |
| --- | --- | --- |
| Проверил |  | А.В. Шелест |
|  | (подпись) |  |
| зачтено |  |  |
|  | (дата защиты) |  |
|  |  |  |
| Выполнил |  | Е.А. Корсаков  гр. 114301 |
|  | (подпись) |  |

Минск, 2024

Цель: *реализация системы аутентификации и авторизации пользователей ПС и механизмов обеспечения безопасности данных*

Ссылка на репозиторий *GitHub*: [BUSIK](https://github.com/D1le1/BUSIK)

Авторизация в программном средстве на Java строится на передаче клиентских данных (логин и хэш пароля) на сервер, где происходит проверка этих данных на соответствие сохраненной информации в базе. Успех зависит от найденного совпадения, иначе сервер возвращает сообщение об ошибке.

Авторизация – это процесс проверки прав доступа пользователя к системе на основе предоставленных данных (учетных записей). Она отличается от аутентификации тем, что фокусируется на разрешении или ограничении доступа к функциям системы.

В данном случае, авторизация базируется на серверной проверке введенных пользователем данных.

**Основные этапы процесса авторизации:**

1. **Ввод учетных данных пользователем:** пользователь вводит логин (в данном случае — номер телефона) и пароль в клиентском приложении. Эти данные не должны храниться или передаваться в открытом виде.
2. **Хэширование пароля:** пароль, введенный пользователем, хэшируется на клиентской стороне перед отправкой на сервер. Хэширование — это необратимый процесс преобразования строки в зашифрованный вид, который используется для хранения паролей. Это важно для защиты данных, так как даже если хэш будет перехвачен, его сложно декодировать.

Пример алгоритмов: SHA-256, bcrypt.

Важно: клиент никогда не отправляет серверу пароль в открытом виде.

1. **Передача данных на сервер:** логин и хэш пароля передаются на сервер через защищенное соединение (например, HTTPS). Это защищает данные от перехвата во время передачи.
2. **Проверка данных на сервере:** сервер принимает запрос и проверяет:
   * существует ли пользователь с указанным логином.
   * совпадает ли хэш пароля, сохраненный в базе данных, с хэшем, полученным от клиента.

Для этого используется следующая логика:

* + сервер находит пользователя по логину.
  + если пользователь существует, сравнивается хэш пароля.
  + если пользователь не найден или хэши не совпадают, авторизация отклоняется.

1. **Ответ сервера:**
   * если данные верны, сервер возвращает положительный ответ, что позволяет пользователю получить доступ к системе.
   * если данные неверны (пользователь не найден или хэши пароля не совпадают), сервер отправляет сообщение об ошибке, например: *"Неверный логин или пароль"*.
2. **Обработка результата авторизации:** Клиентская сторона интерпретирует ответ сервера и либо предоставляет доступ к системе, либо уведомляет пользователя о неудаче.

**Преимущества такого подхода**

1. **Безопасность:** благодаря хэшированию, даже в случае утечки базы данных или перехвата данных при передаче злоумышленник не сможет восстановить оригинальный пароль.
2. **Удобство:** использование номера телефона как логина упрощает идентификацию пользователя.
3. **Гибкость:** такой подход легко масштабируется и интегрируется с дополнительными уровнями безопасности, например, двухфакторной аутентификацией.

**Возможные улучшения системы авторизации**

1. **Соль для хэшей:** Чтобы защититься от атак с использованием радужных таблиц, сервер может использовать "соль" (случайная строка, добавляемая к паролю перед хэшированием).
2. **Двухфакторная аутентификация (2FA):** Дополнительно к логину и паролю можно использовать временные коды или подтверждение через SMS.
3. **Лимит попыток авторизации:** Для предотвращения атак методом перебора можно ограничить количество попыток ввода пароля за определенное время.
4. **Журналирование:** Ведение логов успешных и неуспешных попыток авторизации помогает отслеживать подозрительную активность.

Пример кода обработки авторизации на сервере представлен на рисунке 1.

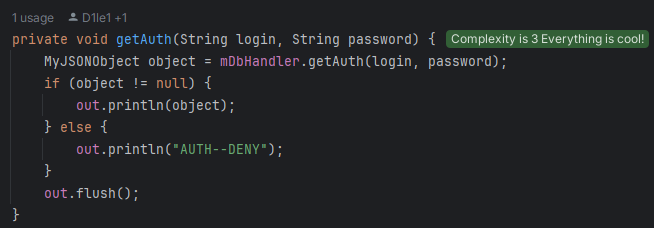


Рисунок 1 – Обработка авторизации на сервере

Пример кода обработки авторизации на клиенте представлен на рисунках 2-3.

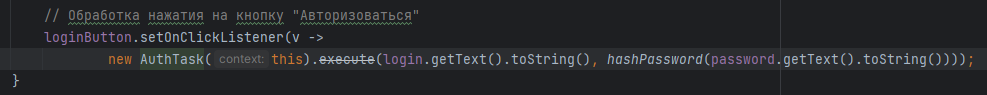


Рисунок 2 – Обработка нажатия кнопки «Авторизоваться»

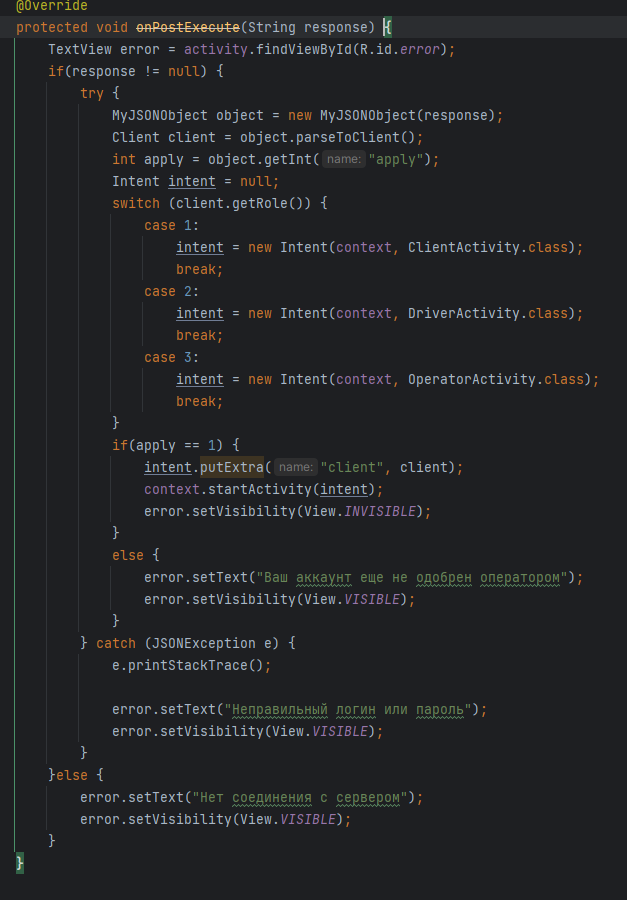


Рисунок 3 – Обработка ответа от сервера о авторизации

Пример кода алгоритма хэширования представлен на рисунке 4.

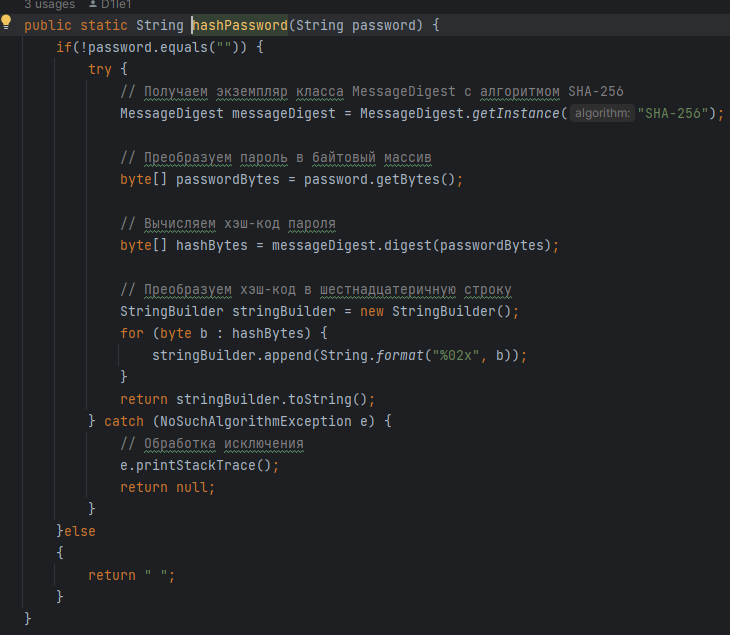


Рисунок 4 – Алгоритма хэширования пароля

Вывод. В рамках практической работы были реализованы системы аутентификации и авторизации, а так же механизмы обеспечения безопасности данных в разрабатываемом программном средстве.