УДК 004.9

ИССЛЕДОВАНИЕ ИЗМЕНЕНИЙ ФОРМЫ УПВ ПРИ ИЗМЕНЕНИЯХ АРТЕРИАЛЬНОГО ДАВЛЕНИЯ

TO STUDY CHANGES IN THE SHAPES OF OLA WITH CHANGES IN BLOOD PRESSURE

С.С. БОГДАН– студент, Институт информационных технологий и радиоэлектроники, кафедра ИСПИ, группа ПРИ-120

А.М. ИЛЛАРИОНОВ– студент, Институт информационных технологий и радиоэлектроники, кафедра ИСПИ, группа ПРИ-120

О.Н. ШАМЫШЕВА – научный руководитель, к.т.н., Институт информационных технологий и радиоэлектроники, кафедра ИСПИ

S.S. BOGDAN – undergraduate, Vladimir state university

A.M. ILLARIONOV – undergraduate, Vladimir state university

O.N. SHAMICHEVA – candidate of technical sciences, Vladimir state university

Аннотация: Разработано приложение, включающее реализацию криптографических алгоритмов с помощью хеш-функций, ориентированное на область создания, хранения и обработки данных пользователей. Описаны архитектура и логика нашего приложения, уделено внимание некоторым важным и ключевым решениям по проектированию и разработке приложения, предоставлено описание классов или их интерфейсов. Включены изображения, отображающие работу программы.

Abstracts: An application has been developed that includes the implementation of cryptographic algorithms using hash functions, focused on the creation, storage and processing of user data. The architecture and logic of our application are described, attention is paid to some important and key decisions on the design and development of the application, and a description of classes or their interfaces is provided. Images showing how the program works are included.

Ключевые слова:хеш-функция, алгоритмы хеширования, sha-1, структура данных, интерфейс.

Keywords:hash function, hash algorithms, sha-1, data structure, interface.

Введение

Хеш-функция— функция, осуществляющая преобразование массива входных данных произвольной длины в (выходную) битовую строку установленной длины, выполняемое определённым алгоритмом. Преобразование, производимое хеш-функцией, называется **хешированием**.

Мы разработали приложение, включающее реализацию алгоритмов хеширования на основе хеш-функций, ориентированное на область обработки и сохранения данных пользователей. Наш проект доступен на GitHub.

Описание работы приложения

Пользователь вводит логин и пароль и выбирает одну из двух функций «Вход» и «Регистрация». В зависимости от этого программа должна выполнить соответствующую логику:

1) Для входа – проверка корректности введенных данных, хеширование данных, поиск и сравнение хешей данных с «базой данных». Если валидация прошла успешно, известить об этом пользователя и выполнить вход в личный кабинет, в котором будут отображаться его данные. Некоторые данные можно будет изменить, а значит также реализоваться соответствующую для этого логику.
2) Для регистрации – проверка корректности введенных данных, хеширование и запись нового пользователя в «базу данных».

Важные нюансы

Из-за ограниченного времени и знаний наша БД будет представлять собой каталог бинарных файлов.

Будем учитывать факт того, что БД может быть достаточно велика, чтобы не помещаться в оперативную память, а значит нужно спроектировать систему так, чтобы она обрабатывала неограниченные объемы данных.

Архитектура приложения

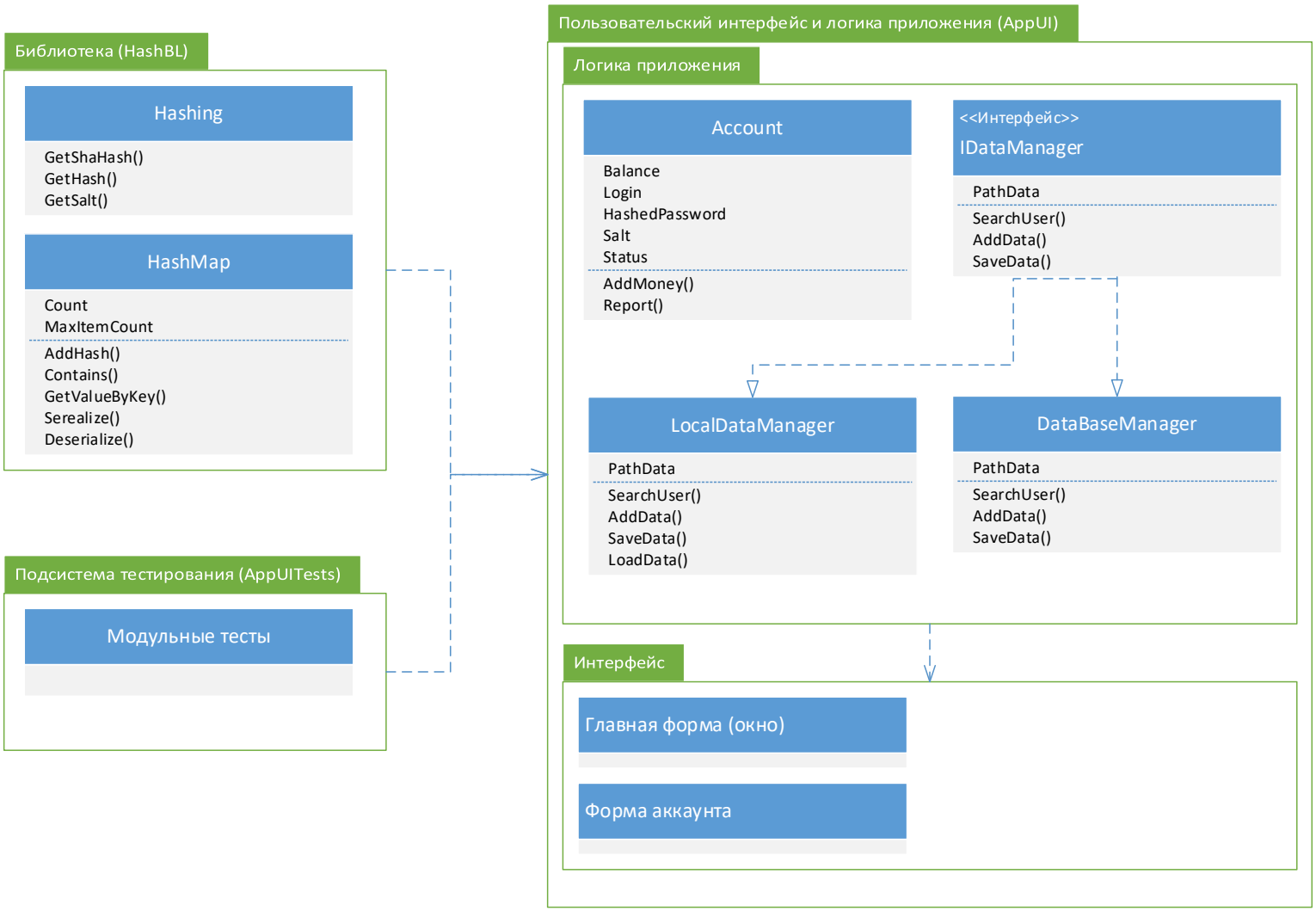

Рис. 1 Схема архитектуры

Хеш библиотека (HashBL)

Класс Hashing

Так как этот класс выступает в роли «функциональной коробочки», то нет смысла создавать экземпляры данного класса, а потому мы сделаем его статическим, это значит, что обращаться ко всем публичным методам, свойствам и полям мы можем без создания экземпляра этой абстракции.

Класс HashMap

Этот класс представляет структуру данных – хеш-таблицу. Хранение данных организовано с помощью словаря, ключом которого является ulong переменная – это будет хеш логина, а значением тип TValue – шаблон, придающий гибкость использования данного класса. В дальнейшей разработке мы решили сделать специальный тип данных для аккаунтов пользователей, объявив Account класс, о котором расскажем чуть позже.

Класс Hashing

Алгоритм хеширования SHA-1

Реализует алгоритм хеширования SHA-1, который преобразует строку в 512 битноесообщение. Входящей строкой является пароль пользователя сканкотанированный с солью, которая генерируется методом, описанным ниже. Сам алгоритм сначала преобразует входящую строку в битовый вид, после в конец строки добавляются биты до тех пор, пока длина не будет равняться 512 битам. Потом последние 64 бита заменяются на двоичное представление длины входящей строки. После с помощью рекуррентной формулы длина битового сообщения увеличивается до 2560 бит. Затем с помощью битовых операций в цикле и пяти заданных констант мы получаем хэш значение.

Алгоритм хеширования через простое число в степени

Реализует алгоритм простого хеширования с использованием простого числа в степени и кода символа. Данный метод нужен только для хеширования логинов, а получившийся хэш будет использоваться в качестве ключа в структуре.

Алгоритм получения соли

Генерирует соль, т. е. строку фиксированной длины со случайными символами. Эта строка по ходу алгоритма прибавится к паролю перед его хешированием. Соль необходима для усложнения определения прообраза хэш-функции.

Вспомогательные методы для хеширования SHA-1

Первый метод реализует циклический сдвиг на определенное количество бит. Второй метод преобразует строку, которая является паролем, в список из

беззнаковых интеджеров (*uint*) длиной в 80 символов. Этот список будет нужен по ходу алгоритма хеширования пароля.

Класс HashMap<TValue>

Методы сохранения данных в файл

Сохранение данных в файл определено методом *Serealize(stringpath).* Данный метод при помощи встроенного метода сохраняет словарь с данными в файл. Выгрузка данных из файла определено методом *Deserealize(stringpath).* Данный метод также при помощи встроенного метода инициализирует словарь с данными.

Добавление значений в структуру

Добавлениеданныхопределенометодом *AddHash(ulonghash, TValuevalue).* Данный метод также должен проверять существование такого хеша, а также следить за тем, чтобы не произошло переполнение структуры данных.

Получение значения по ключу

За данную операцию отвечает метод *GetValueByKey(ulongkey).* Для защиты от ошибок нам все равно необходимо проверить входные данные, т.е. определить, есть ли в коллекции вводимый ключ. Если заданный ключ имеется в коллекции, то возвращается значение, привязанное к этому ключу, иначе возвращается значение *default* для типа *TValue*.

Пользовательский интерфейс и логика приложения (AppUI)

Класс Account

Объекты этого класса будут представлять пользователей системы. Класс имеет *[Serializable]* атрибут, так как его объекты будут храниться в структуре данных в качестве *TValue* значения.

Класс LocalDataManager

Этот класс реализует обработку «локальной базы данных», имплементируя интерфейс *IDataManager.* Мы уже говорили, что сделали хранение данных в виде бинарных файлов. Однако думая о масштабируемости и практике, мы понимаем, что данному типу приложения уместно сделать настоящую базу

данных, которая возможно будет располагаться и не локально. Поэтому при проектировании был создан интерфейс управления данными (*IDataManager*), имплементируя который можно распределить управление данных на локальное (*LocalDataManager*) и глобальное (*DataBaseManager*). Мы не будем напрямую обращаться к классам *LocalDataManager* и *DataBaseManager*, вместо этого создадим *IDataManager* «контейнер», это позволит сильно сократить кол-во изменений в коде - одной строки будет достаточно, причем сделать это можно даже программно.

Пользовательский интерфейс

Кнопка «Войти» (Располагается на главной форме)

Срабатывает при нажатии на кнопку «Войти». Сначала происходит проверка логина и пароля на корректность, затем определяется, есть ли такой пользователь в базе данных и также сравниваются хэшы паролей и если всё совпадает, то приложение разрешает вход - пользователь переходит на форму аккаунта.

Кнопка «Зарегистрироваться» (Располагается на главной форме)

Срабатывает при нажатии на кнопку «Зарегистрироваться». Сначала также происходит проверка логина и пароля на корректность, затем определяется есть ли такой пользователь в базе данных и если такого нет, то происходит добавление нового пользователя в «базу данных».

Кнопка «Зачислить средства» (Располагается на форме аккаунта)

После проверки входных данных зачисляет введенное пользователем значение на баланс.

Работа приложения

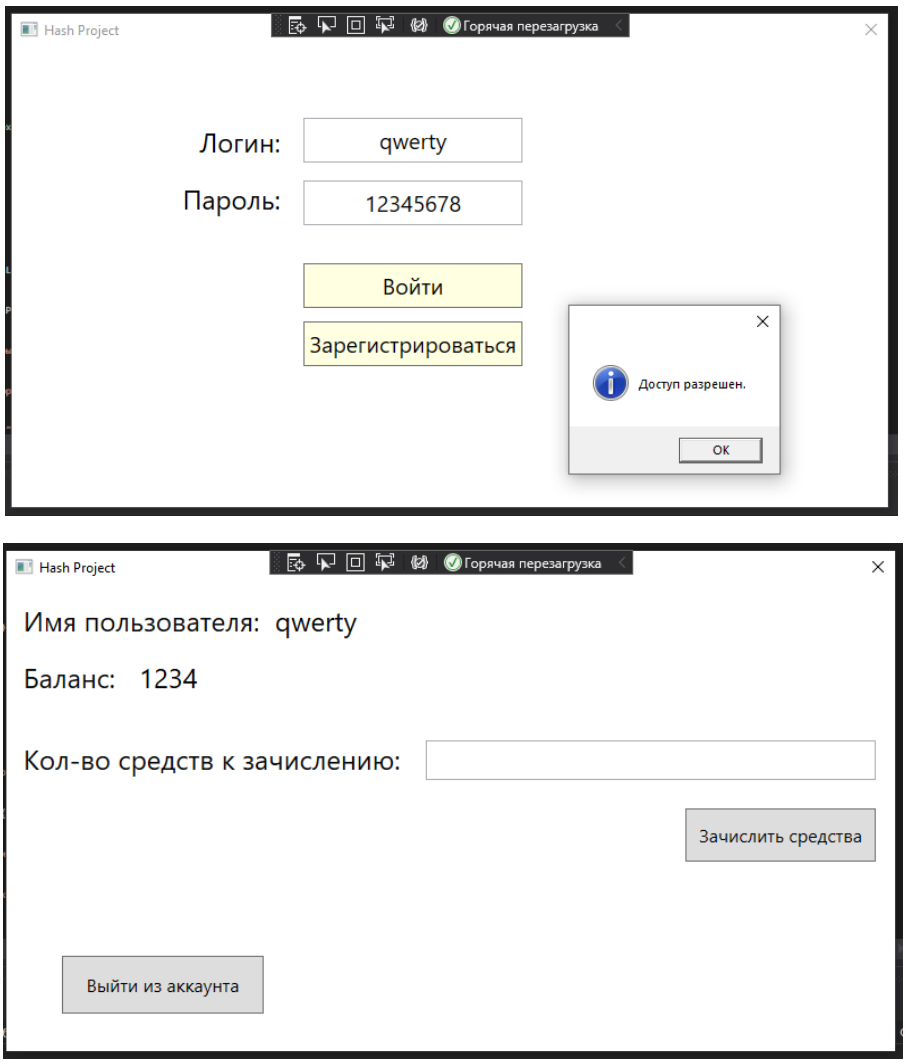

рис.3.1–3.3 Работа приложения