*Piczle*

Developed by

Antonius Cezar Hegyes

Contents

[Purpose 2](#_Toc382496872)

[Concept of work 2](#_Toc382496873)

[Accessibility 3](#_Toc382496874)

[Security 3](#_Toc382496875)

[Challenges along the way 4](#_Toc382496876)

[Future plans 5](#_Toc382496877)

[Other possible uses 5](#_Toc382496878)

# Technologies used

* C# - **W**indows **P**resentation **F**oundation (WPF) Project
* NoSQL Database – Dbreeze
* ESENT – Windows built-in database
* sCrypt – library with an improved implementation of the bCrypt security algorithm

# Purpose

Piczle is a digital didactic solution: a software able to grasp the attention of young children while teaching them history / social issues / any other thing. Each lesson follows a certain flow and a lot of vital information can be transmitted through visual information. Unfortunately, most of what we see is not retained by our brains, which will forget over 90% of what we saw just 3 days ago.

Piczle is a solution for that! Given that children are easily impressionable, but that they fail to pay attention for long period of times, Piczle offers them a mean to switch their main focus fast enough for them not to lose concentration on the topic and also slow enough for them to grasp the taught concepts. This way, the visual image given by Piczle will be retained by their young brains!

# Concept of work

Basically, what Piczle does is taking an image of any sort and it turns in into a puzzle. Upon completion of the puzzle, the resolver will receive a custom message and the application will display a short description of what the image depicts.

The children will remember the solving of the puzzle and the “reward” (image) they received afterwards, thus embedding the conclusion of the lesson into their minds. The power of Piczle lies in the fact that one can change between two different types of puzzles – *Separate pieces* or *Scrambled image***–** and also select the number of pieces which the puzzle should be composed of. This way, each puzzle will be unique in more ways than just one.

Each session is based on a username/password combination, thus allowing every player to continue from where he/she left off last time. The user is free to either solve the given, default puzzles or to create categories and upload pictures of his own, which offers a teacher and even user full control over the content available on the account.

The program even keeps track of a score. Upon completing a puzzle, one receives a certain boost of points based on the completion time – the faster the user finished the puzzle, the more points he/she gets. If one grows tired of finding a solution, one must only click the ‘Solve puzzle’ button and the computer will do the rest of the work (this will cost 5 points of the score so that the scholars do not grow lazy into solving it because of their instinct of competition).

# Accessibility

Piczle is already available on two languages, but in order to add new ones, no more than an hour is needed. That’s because the text used is saved in a SQL Compact Edition 4.0 Database, with each Window having its own table with two columns: name of the object and its content. At runtime, the content of the database holding the language wished by the user will be loaded in the RAM for fast access. For security reasons, the databases are password-protected.

In order to store your data, Piczle uses NoSQL technology. NoSQL means total flexibility and ease of scaling out. It is the exact opposite of RDBMS (**R**elational **D**ata**B**ase **M**anagement **S**ystem) like MySQL for example. In a RDBMS environment there are databases and each database has tables. In turn, tables have columns and each column has a data type which it can store (number, string, time etc.). Each row of the table must comply with those rules. Each time a new functionality is added or the code logic is changed, database modifications are bound to happen. This results in a huge amount of work every time.

The user’s data (namely the custom pictures and the related information) stays available thanks to a local NoSQL Database – [Dbreeze](http://dbreeze.codeplex.com/). Dbreeze is an [ACID](http://en.wikipedia.org/wiki/ACID) compliant embedded-database for .NET platforms and [mono](http://www.mono-project.com/Main_Page) compatible. For keeping track of users using the program on a certain computer (name, high scores, password etc.), a [PersistentDictionary](http://managedesent.codeplex.com/wikipage?title=PersistentDictionaryDocumentation&referringTitle=Documentation) of the Windows-embedded database [ESENT](http://blogs.msdn.com/b/windowssdk/archive/2008/10/23/esent-extensible-storage-engine-api-in-the-windows-sdk.aspx) has been implemented.

# Security

Safety is a major concern in the world of IT. Privacy is a basic principle of our society and the right of every human being. Unfortunately, not everyone respects this and there could be people out there who might wish to put their hands on the user’s data. Although extensive security for a program dedicated to children might seem extraneous, it has been implemented at least for the sake of programming and practice.

Accessing a user’s data can be achieved in one of two ways: either manage to get the password and log in normally or hack into Dbreeze and get the data regarding the specific user they are searching for. If the user’s password is „1234abcd” or „password” there is nothing to be done but otherwise, a programmer needs to make sure that any data in the database is plain rubbish for any malicious attacker in whose hands it might get.

For these reasons exactly, Piczle has implemented some of the foremost cryptographic algorithms in the world. All data regarding user X is encrypted with the [Rijndael algorithm](http://searchsecurity.techtarget.com/definition/Rijndael) and a 256bit key derived from the [Rfc2898DeriveBytes](http://msdn.microsoft.com/en-us/library/system.security.cryptography.rfc2898derivebytes.aspx) function – a special pseudo-hashing algorithm with a given iteration count of 10000 to further increase safety against precomputed attacks. To crack this encryption alone, one would need tremendous power.

Rfc2898DeriveBytes needs itself a key and a salt. The salt is a unique string for each user which makes sure that two texts won’t have the same encrypted string to avoid [Dictionary attacks](http://en.wikipedia.org/wiki/Dictionary_attack) and it is public (in this case, the Guid of each user). The key is a special output about I which I will discuss later.

Trying to brute-force-wise guess the key is overkill because it is represented by the Base64String of 128 bytes which means more than 30 characters. A Base64String of 8 characters has 218 trillion possible combinations (62.3 times more than a 7 characters one). A [Class F computer](http://www.lockdown.co.uk/?pg=combi#classF) needs 60 hours to try each of them, whereas a Class D (average Dual-core PC) needs 253 days! If someone would try to decrypt the data this way though, they would have to do the job for each user individually because the output is based on the user’s password. Through the addition of Salts, that output will be different even if the password of two users match.

The other approach would be to brute-force-wise guess the user’s password and create that unique output – it is almost certain the user doesn’t have a 30 characters password. This is not going to be easy either because the password is ran through a [sCrypt](http://xato.net/application-security/yes-use-bcrypt-and-scrypt/) algorithm first. sCrypt is built on top of [bCrypt](http://en.wikipedia.org/wiki/Bcrypt) and adds a salt implementation (again, the Guid) and lets the programmer choose the amount of memory needed to compute the string. Not to mention, they are way more secure than [Hashing](http://en.wikipedia.org/wiki/SHA-2).

Where does their power lay in? Slowness! Hashing is fast, bCrypt is way slower (0.3 seconds computing of a string of 30 characters is very slow) and sCrypt set to use 8MB of RAM needs 3 seconds – 10 times the time of bCrypt – on a Class D PC. If a user’s password is made out of 8 characters and the hacker checks for those 8 chars out of an 62 characters database (assuming that the user could have chosen anything available on a standard US-keyboard as part of his password), those 60 hours needed by a Class F computer just got multiplied by a lot:

1. 218 trillion combinations in 60 hours means 9.9 \* 10-10 sec/combination, but let’s round up to 1 \* 10-11 sec/combination
2. Now we need 3 seconds/combination
3. 218 trillion combinations in 60 hours just went up to 2054794520 years, which will have to be repeated for each user separately because of the Password-Based-Encryption

The user won’t notice those 3 seconds delay, but the hacker will have a really, really hard time decoding the data/brute-forcing the password. Data is secure with Piczle!

The easiest way to get a user’s password is from the user’s computer directly. For functionalities such as ‘Forgot my password’, the raw password needs to be *known* by that PC. The password is not going to be saved in its raw form, but shall be encrypted first using the [Windows DPAPI](http://en.wikipedia.org/wiki/Data_Protection_API). This way, only that computer logged in on that user profile will be able to decrypt the password.

Nevertheless, if the hacker has access to such a computer, he would still need to get access to the original code and write a program that will reverse engineer the code. Quite possible, but really not worth it and still extremely hard to do.

# Challenges along the way

Apart from the security, which required extensive research and various designs until reaching its present form, the project has risen a few other problems which needed to be overcome. The biggest one of them was making the transition of the pieces in the “Scrambled image” mode to perform smoothly. This was achieved by working on a different thread than the main one and, at each step, forcing the rendering thread to keep up with the changes in the structure. The use of watermarked input boxes also necessitated the implementation of a service based on adorner layers. Nevertheless, optimizing the visual appearance to be friendly and interesting for children has posed more of a challenge than I had expected.

# Future plans

The final version of the program should include some more efficient algorithms for the ‘Solve puzzle’ function, online multiplayer capabilities, and the ability to create virtual classes and connect members of a class to a session to further facilitate learning.

# Other possible uses

Given the level of security at its disposal, Piczle can also be used to transmit messages which should not be available to everyone. A very basic example to exemplify this can be the desire to express feelings for someone: prepare a picture with a written text and ask the person supposed to read it to solve the puzzle. Once uploaded into the account and deleted from the computer, the picture can no longer be seen by someone who does not control your password.

After implement the multiplayer capabilities, this will be achievable even easier by simply sending someone a request to solve a picture you sent them. Basically, anything which requires to convey a message can be achieved by Piczle, even in its current form!