Министерство образования и науки Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра математической
кибернетики и компьютерных наук

WEB-МЕНЕДЖЕР ДЛЯ ХРАНЕНИЯ ПАРОЛЕЙ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Рафикова Рустана Ринатовича

Научный руководитель
доцент, к. ф.-м. н. _____ Ю. Н. Кондратова

Заведующий кафедрой
к. ф.-м. н. _____ С. В. Миронов

Саратов 2018

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Пароли являются основой для обеспечения информационной безопасности. Они используются в качестве первой линии защиты для обеспечения безопасности электронной информации, сетей, серверов, устройств, учетных записей, баз данных, файлов и так далее. Каждый пароль должен быть уникален, длинными и сложными, состоящим из цифр, букв смешанного алфавита и специальных символов. Он также не должен быть словами или именами, которые могут быть связаны с их владельцем. Наконец, пароли должны часто меняться.

Такие принципы к выбору паролей приводят к тому, что их очень сложно запоминать. Приложения для управления паролями являются решением этой проблемы. Такие приложения должны отвечать требованиям безопасного управления паролями, помнить неограниченное количество паролей, быть простыми в использовании. Хотя есть и другие способы запоминать и отслеживать пароли, ни один из них не предлагает удобство, которое может предоставить подобное приложение.

Целью данной работы является разработка web-менеджера для индивидуального и коллективного хранения паролей средствами языка программирования Python. Для достижения этой цели необходимо решить следующие задачи:

1. создание и настройка базы данных;
2. рассмотрение алгоритмов шифрования для разработки методов хранения паролей в базе данных;
3. разработка web-интерфейса приложения;
4. реализация системы коллективного доступа к паролям.

Бакалаврская работа содержит следующие разделы:

— введение;
— инструменты для разработки web-менеджера для хранения паролей;
— разработка web-менеджера для хранения паролей;
— заключение.

1 Основное содержание работы

Инструменты для разработки web-менеджера для хранения паролей. Данная глава описывает существующие менеджеры для хранения паролей, технологии и методики для реализации web-менеджеров, а также управление доступом к данным.

В первой части данной главы были рассмотрены менеджеры для хранения паролей.

Существующие приложения, являющиеся менеджерами хранения паролей, делятся на две категории: локальные и удаленные, что подтверждается в источнике в [1]. Их отличие заключается в том, что в локальных все пароли хранятся на устройстве пользователя, а в глобальных — на сервере. Первый подход более надежен, а второй более комфортен за счет автоматической синхронизации на всех устройствах.

Далее рассмотрены примеры существующих менеджеров, как локальных, так и глобальных.

В качестве локальных приведены следующие менеджеры:

— KeePass;
— RoboForm;
— 1Password.

В качестве удаленных приводятся следующие менеджеры:

— Passpack;
— Splikity;
— LastPass.

Для всех менеджеров была проведена сравнительная характеристика по их недостаткам и достоинствам.

Во второй части главы были рассмотрены технологи иметодики для реализации web-менеджера. В частности, описаны модели, архитектура web-приложений и существующие инструменты для разработки.

Выбранные для рассмотрения типы моделей web-приложений приведены ниже. Данная классификация приведена из источника [2].

— один web-сервер (с базой данных) — самая простая и наиболее рискованная модель, где одна база данных является частью единственного сервера web-приложения. Если сервер станет неисправным, то и web-приложение перестанет функционировать.

— несколько web-серверов и одна база данных. Идея заключается в том, что web-сервер не должен хранить какие-либо данные: даже когда он получает информацию от клиента, web-сервер обрабатывает ее, записывает данные в базу данных и забывает об этом. Такой принцип также известен под названием «архитектура без состояния».

— несколько web-серверов и несколько баз данных. Применяя такую модель, есть два варианта: базы данных хранят идентичные данные или данные равномерно распределены между ними. В первом случае обычно требуется не более двух баз данных. Когда одна из них выходит из строя, другая может заменить ее. Поскольку данные не реплицируются во втором случае, некоторые данные могут стать временно недоступными, если одна из многих баз данных аварийно завершает работу. Тем не менее эта модель считается наиболее отказоустойчивой: ни web-серверы, ни базы данных не имеют единственных точек отказа.

Используя источник [3] были выделены следующие типы архитектуры web-приложений:

— базовая архитектура web-приложения. Согласно самой первой и базовой архитектуре web-приложений, сервер, состоящий из логики построения web-страницы и бизнес-логики, взаимодействует с клиентом, отправляя всю HTML-страницу. Чтобы увидеть обновление, пользователю необходимо полностью перезагрузить страницу или, другими словами, клиенту необходимо отправить запрос на сервер и снова загрузить весь код.

— widget архитектура. В этом типы архитектуры логика построения web-страницы заменяется web-сервисами, и каждая страница на клиенте имеет отдельные объекты, называемые виджетами. Отправляя AJAX запросы на web-сервисы, виджеты могут получать куски данных HTML или JSON и отображать их без перезагрузки всей страницы. Благодаря обновлениям виджетов в режиме реального времени этот тип более динамичен, удобен для отображения на мобильных устройствах и почти столь же популярен, как и следующий тип.

— архитектура одностраничного web-приложения. Это самая современная и популярная архитектура web-приложений, где одна страница загружается только один раз. На стороне клиента эта страница имеет дополнительный JavaScript уровень, который может свободно взаимодей-

ствовать с web-службами на сервере и, используя данные из web-служб, делать обновления в реальном времени сам по себе.

В третьей части данной главы были рассмотрены методы хранения паролей. Выделен самый небезопасный способ хранения — пароль для каждого пользователя хранится в базе данных без шифрования, хеширования или любых преобразований. Среди других способов были рассмотрены хеширование паролей и использование криптографии (добавление криптографической «соли» к паролю, прежде чем передавать его через хеш-функцию; как правило, это объединение случайной строки в начало или конец исходного пароля). Согласно источнику [4] важным вопросом при использовании хеширования является выбор алгоритма хеширования, поэтому были рассмотрены следующие алгоритмы:

— MD5;
— SHA-1;
— PBKDF2;
— bcrypt.

Также были рассмотрены средства защиты данных в языке программирования Python. В частности, приведен механизм обеспечения надежного хранения паролей фреймворка Django.

Последняя часть данной главы содержит в себе описание основных понятий управления доступом к данным, причем особое внимание уделяется ролевому управлению доступом.

Разработка web-менеджера для хранения паролей. Было разработано web-приложение для хранения паролей.

При разработке приложения были использованы:

— `Python` v3.5 — язык программирования серверной части;
— `Django` v2.0.5 — для создания веб приложения и настройки процессов передачи запросов и доступа к базе (официальный сайт [5]);
— `Django-mptt` v0.9.0 — для создания древовидной структуры в базе (официальный сайт [6]);
— `Pillow` v5.1.0 — для обработки загружаемых изображений (официальный сайт [7]);
— `JavaScript` — язык программирования клиентской части;
— `Jquery` v3.3.1 — для написания и обработки событий (официальный сайт [8]);

— `JQuery contextMenu` v2.6.4 — для создания собственного контекстного меню (официальный сайт [9]).

Приложение состоит из следующих компонентов:

- `Diploma` — отвечает за настройки приложения и url карту;
- `authentication` — отвечает за аутентификацию пользователей;
- `registration` — отвечает за регистрацию пользователей;
- `registration` — отвечает за регистрацию пользователей;
- `passwords` — отвечает за хранение паролей и директорий в базе, а также за их использование;
- `static` — отвечает за статичные файлы, в свою очередь подразделяется на:
  - `css` — каскады стилей для html;
  - `images` — статичные изображения (не загружаемые пользователем);
  - `js` — скрипты для клиентской части;
- `teams` — отвечает за хранение коллективов и их настройки ролевого доступа в базе данных;
- `templates` — хранит шаблоны html страниц;
- `users` — отвечает за хранение пользователей в базе.

При входе в приложение пользовтель попадает на стартовую страницу, на которой он может осуществить вход в свой аккаунт или зарегистрироваться. За интерфейс данной страницы отвечает `general/templates/start page.html`. Сами формы входа и регистрации не задаются вручную внутри html. Вместо этого они создаются средствами `Django`.

Был реализован функционал восстановления пароля. При нажатии на ссылку «Забыли пароль?» открывается страница, на которой требуется ввести свой адрес электронной почты, указанный при регистрации. После ввода пользователю на почту приходит сообщение, в котором содержится автоматически сгенерированная ссылка для восстановления пароля. Данная ссылка работает до тех пор, пока пользователь не перейдет по ней и не поменяет свой пароль. После того, как пользователь перешел по ссылке, он оказывается на странице, на которой требуется ввести новый пароль. При успешном вводе ссылка перестает быть активной.

При успешной аутентификации пользователь попадает на страницу личных паролей. Данная страница разделяется на несколько частей:

— «Шапка» сайта. Она может использоваться на нескольких страницах;
— Дерево каталогов (папок);
— Таблица, содержащая сохраненные пароли пользователя.

Дерево каталогов необходимо для предоставления пользователям возможности хранения паролей в разных каталогах. Такая древовидная структура обеспечивает удобство пользования. В данном дереве реализовано контекстное меню. При нажатии правой кнопки мыши пользователю предоставляется возможность осуществить следующие действия: добавить каталог, переименовать каталог, удалить каталог. Для обработки вызова контекстного меню были написаны следующие функции:

— `get_opt_element` — вспомогательная функция возвращающая элемент, от которого контекстное меню было вызвано;
— `get_opt_id` — вспомогательная функция возвращающая идентификатор элемента, от которого контекстное меню было вызвано;
— `check_if_root` — проверка того, является ли папка корневой (для нее недоступно действие удалить);
— `add_dir` — операция добавления нового каталога;
— `rename_dir` — функция переименования каталога;
— `delete_dir` — операция удаления каталога;
— `edit_password` — операция редактирования пароля;
— `delete_password` — операция удаления пароля.

При каждом из действий контекстного меню сначала происходит проверка того, что действие закончено, а также что необходимо обновление базы данных. Если проверка прошла успешна, то отправляется POST запрос на соответствующий адрес:

— `/passwords/personal/add_dir` — для добавления папки;
— `/passwords/personal/edit_dir` — для переименования;
— `/passwords/personal/delete_dir` — для удаления.

Также была реализована возможность перемещения каталога внутрь другого посредством перетаскивания (функционал основан на событиях html5).

В таблице паролей также реализованы функция перетаскивания и контекстное меню. В контекстном меню только 2 записи: редактировать и удалить. Для добавления пароля в нижней строке таблицы закреплена специальная форма. У полей логин и пароль также добавлена кнопка копирования,

которая копирует значение в буфер обмена. У пароля добавлена кнопка отобразить для раскрытия значения пароля. Повторное нажатие на эту кнопку уберет эффект.

В приложении была реализована возможность коллективного хранения паролей. Этот подраздел открывается при нажатии на ссылку «Команды» в шапке сайта. При переходе в этот раздел сайта отображается страница, на которой изображены команды, в которых пользователь уже состоит, и форма для создания новой команды. Каждая команда имеет собственную настройку ролей, что позволяет гибко настраивать систему под различные нужды. Возможность выполнения каждого из действий определяется ролью пользователя.

В качестве системы управления базами данных используется `sqlite`. База данных содержит следующие таблицы:

- `users_customuser` — информация о пользователях (имя пользователя, пароль, электронная почта, телефон, корневая директория для личных паролей);
- `teams_team` — информация о группах (название, корневая директория для паролей);
- `teams_member` — информация о членах групп(идентификатор группы, роль, идентификатор пользователя);
- `teams_role` — информация о ролях (название, идентификатор группы, наличие прав записи, иконка);
- `passwords_directories` — информация о директориях (название, родительская директория, индекс дерева, которому она принадлежит, секретный ключ);
- `passwords_personal` — информация о личных паролях (название, логин, пароль, иконка, url, комментарий, идентификатор владельца, дата создания);
- `passwords_collective` — информация о групповых паролях(название, логин, пароль, иконка, url, комментарий, идентификатор группы-владельца, идентификатор создателя и последнего редактора, дата и время создания и последнего редактирования).

Для взаимодействия приложения с таблицами были написаны следующие модели:

— В файле `users/models.py`:
  - `CustomUser` — класс, отвечающий за пользователей. Он унаследован от класса `AbstractUser` с добавлением полей `root_dir`, `phone`, `settings` и свойства `teams`. Также были переопределены поле `email` и расширены методы `clean_fields` и `save`.
  - `Settings` — класс, отвечающий за настройки пользователей. В нем содержатся следующие поля: `email_confirmed`, `phone_confirmed`, `two_step_auth`, `days_before_deleting`, `email_notifications`.
— В файле `teams/models.py`:
  - `Team` — класс, отвечающий за группы, в котором содержатся поля `title` и `root_dir`. В нем также был расширен метод `save`.
  - `Member` — класс, отвечающий за членов групп. В нем определены поля `user`, `role` и `team`, а также переопределено строковое представления для улучшения отображения.
  - `Role` — класс, который отвечает за роли членов в группах.В нем определены следующие поля: `title`, `team`, `write_permission`, `icon`.
— В файле `passwords/models.py`:
  - `Directories` — класс, отвечающий за директории. Он был унаследован от `MPTTModel`, с добавлением полей `secret_key`, `is_deleting`, `title`, `parent`.
  - `Passwords` — абстрактный класс, отвечающий за пароли. В нем содержатся следующие поля: `title`, `login`, `password`, `url`, `comment`, `icon`, `create_date`, `directory`. Для него были также переопределены мэнеджер и набор запросов. Был расширен метод `save` для сужения размера входного изображения и шифровки изображения.
  - `Personal` — класс, отвечающий за личные пароли. Он расширяет класс `Passwords` добавлением поля `owner`.
  - `Collective` — класс, отвечающий за групповые пароли. Он наследует класс `Passwords` и содержит поля `team`, `creator`, `last_editor`, `last_edit_time`.

В базе данных пароли не хранятся в открытом виде. Вместо этого они шифруются и дешифруются при каждом добавлении и запросе соответственно.

Алгоритм шифрования состоит из следующих шагов:

1. Получаем секретных ключ целевой директории, хранящийся в базе данных.
2. На основе ключа директории создается объект класса `Signer`. Данный класс находится в модуле `django.core.signing`.
3. Шифруем пароль полученным объектом.
4. Переводим получившееся выражения в специальную кодировку.

Дешифрование производится следующим образом

1. Переводим криптограмму из кодировки, используемой при шифровании.
2. Получаем секретных ключ целевой директории, хранящийся в базе данных.
3. На основе ключа директории создается объект класса `Signer`. Данный класс находится в модуле `django.core.signing`.
4. Дешифруем пароль полученным объектом.

Разработанное приложение имеет ряд преимуществ и недостатков, по сравнению с современными аналогами.

Преимущества:

— приложение полностью бесплатное;
— гибкая настройка ролевого доступа;
— открытый исходный код;
— возможность запуска в локальной сети.

Недостатки:

— интерфейс не адаптирован под мобильные устройства;
— отсутствие интеграции в браузер;
— отсутствие двухэтапной аутентификации.

ЗАКЛЮЧЕНИЕ

В ходе написания выпускной квалификационной работы были изучены модели построения web-приложений, рассмотрены способы хранения паролей и возможности управления доступом к данным. Был разработан web-менеджер хранения паролей на языке программирования Python с использованием фреймворка Django, а также был проведен сравнительный анализ с уже существующими менеджерами. Менеджер предоставляет возможность хранения как личных, так и коллективных паролей. Для удобства пользователя имеется возможность структурировать пароли по категориям, например социальные сети, почта и т. д. В приложении также реализовано восстановление пароля аутентификации в web-менеджер с помощью почты. При коллективном хранении паролей предоставляются гибкие настройки ролевого доступа. Добавление новых участников происходит на основе токенов.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Хранение паролей в базе данных [Электронный ресурс]. — URL: http://ekimoff.ru/74/ (Дата обращения 14.05.2018). Загл. с экр. Яз. рус.

2 Web application architecture: Components, models and types [Электронный ресурс]. — URL: https://www.scnsoft.com/blog/web-application-architecture (Дата обращения 01.06.2018). Загл. с экр. Яз. англ.

3 *Эспозито, Д.* Разработка современных веб-приложений. Анализ предметных областей и технологий / Д. Эспозито. — Вильямс, 2017.

4 Хэш-алгоритмы [Электронный ресурс]. — URL: https://habr.com/post/93226/ (Дата обращения 28.05.2018). Загл. с экр. Яз. рус.

5 The Web framework Django [Электронный ресурс]. — URL: https://www.djangoproject.com/ (Дата обращения 6.05.2018). Загл. с экр. Яз. англ.

6 Django MPTT documentation [Электронный ресурс]. — URL: https://django-mptt.readthedocs.io/en/latest/ (Дата обращения 8.05.2018). Загл. с экр. Яз. англ.

7 Pillow (PIL Fork) [Электронный ресурс]. — URL: https://pillow.readthedocs.io/en/5.1.x/ (Дата обращения 17.05.2018). Загл. с экр. Яз. англ.

8 Jquery [Электронный ресурс]. — URL: https://jquery.com/ (Дата обращения 20.05.2018). Загл. с экр. Яз. англ.

9 jQuery contextMenu [Электронный ресурс]. — URL: https://swisnl.github.io/jQuery-contextMenu/ (Дата обращения 25.05.2018). Загл. с экр. Яз. англ.