**1. Импорт модулей**

python

import sys

import random

import string

import json

from PyQt5.QtWidgets import (QApplication, QWidget, QLabel, QLineEdit, QPushButton, QVBoxLayout, QMessageBox,

QTableWidget, QTableWidgetItem, QHeaderView, QInputDialog)

from PyQt5.QtGui import QFont, QPalette, QColor, QIcon

from PyQt5.QtCore import Qt

* **sys**: модуль для взаимодействия с системными операциями, например, для завершения программы.
* **random** и **string**: используются для генерации случайных паролей.
* **json**: используется для сохранения и загрузки паролей в формате JSON.
* **PyQt5.QtWidgets**: это набор классов для создания элементов графического интерфейса (окна, кнопки, текстовые поля и т.д.).
* **PyQt5.QtGui** и **PyQt5.QtCore**: дополнительные модули для управления цветами, иконками и другими графическими элементами.

**2. Функция генерации паролей**

python

def generate\_password(length=12):

characters = string.ascii\_letters + string.digits + string.punctuation

password = ''.join(random.choice(characters) for i in range(length))

return password

* **generate\_password**: функция для генерации случайного пароля длиной 12 символов.
  + **string.ascii\_letters + string.digits + string.punctuation**: строка, содержащая буквы, цифры и специальные символы.
  + **random.choice**: выбирает случайный символ из строки.
  + **''.join()**: объединяет выбранные символы в строку, которая и будет паролем.

**3. Класс PasswordManager**

Этот класс отвечает за управление паролями (загрузка, сохранение, добавление паролей).

python

class PasswordManager:

def \_\_init\_\_(self, filename='passwords.json'):

self.filename = filename

self.passwords = self.load\_passwords()

* **filename**: имя файла, в котором будут храниться пароли.
* **passwords**: словарь для хранения всех паролей, который загружается с помощью метода **load\_passwords**.

**Загрузка паролей:**

python

def load\_passwords(self):

try:

with open(self.filename, 'r') as f:

data = json.load(f)

if isinstance(data, dict): # Проверяем, является ли объект словарем

return data

else:

return {}

except FileNotFoundError:

return {}

* **load\_passwords**: загружает пароли из файла JSON. Если файла не существует, возвращает пустой словарь.

**Сохранение паролей:**

python

def save\_passwords(self):

with open(self.filename, 'w') as f:

json.dump(self.passwords, f)

* **save\_passwords**: сохраняет текущие пароли в файл JSON.

**Добавление пароля:**

python

def add\_password(self, service, username, password):

self.passwords[service] = {"username": username, "password": password}

self.save\_passwords()

* **add\_password**: добавляет новый пароль для указанного сервиса. Пароль сохраняется вместе с логином. После добавления пароля вызывается метод **save\_passwords** для сохранения изменений.

**4. Класс PasswordStorageApp**

Это основной класс приложения, который отвечает за графический интерфейс и функциональность работы с паролями.

python

class PasswordStorageApp(QWidget):

def \_\_init\_\_(self, master\_password):

super().\_\_init\_\_()

self.master\_password = master\_password

self.password\_manager = PasswordManager()

self.init\_ui()

* **master\_password**: мастер-пароль, который пользователь вводит для доступа к приложению.
* **password\_manager**: экземпляр класса **PasswordManager**, который управляет паролями.

**Инициализация интерфейса:**

python

def init\_ui(self):

self.setWindowTitle("Password Storage by HSE")

self.setGeometry(100, 100, 500, 400)

self.setWindowIcon(QIcon("path\_to\_icon.png"))

* **init\_ui**: инициализирует окно с заголовком и значком.

**Установка стилей:**

python

self.setStyleSheet("""

QWidget {

background-color: #e6f7ff;

}

QLabel {

color: #333;

font-size: 15px;

font-weight: bold;

}

QLineEdit {

padding: 10px;

border: 2px solid #5d8bb8;

border-radius: 5px;

font-size: 14px;

background-color: #ffffff;

}

QPushButton {

background-color: #5d8bb8;

color: white;

font-size: 16px;

padding: 12px;

border-radius: 10px;

}

QTableWidget {

background-color: #ffffff;

color: #333;

font-size: 14px;

border: 2px solid #5d8bb8;

border-radius: 5px;

}

QHeaderView::section {

background-color: #5d8bb8;

color: white;

padding: 4px;

font-size: 14px;

}

""")

* Устанавливаются стили для всех элементов интерфейса. Цветовая схема включает светло-голубой и белый цвета. Стиль кнопок и таблицы настраивается индивидуально.

**Таблица для отображения паролей:**

python

self.password\_table = QTableWidget(self)

self.password\_table.setColumnCount(3)

self.password\_table.setHorizontalHeaderLabels(["Сервис", "Логин", "Пароль"])

self.password\_table.horizontalHeader().setSectionResizeMode(QHeaderView.Stretch)

* **password\_table**: это таблица, которая отображает все сервисы, логины и пароли.

**Поле поиска:**

python

self.search\_label = QLabel("Поиск по сервису:")

self.search\_input = QLineEdit(self)

self.search\_input.setPlaceholderText("Введите название сервиса для поиска")

self.search\_input.textChanged.connect(self.search\_service)

* **search\_input**: поле для ввода текста, в котором пользователь может искать сервисы по их названию.

**Кнопка "Добавить сервис":**

python

self.add\_button = QPushButton("Добавить сервис")

self.add\_button.clicked.connect(self.add\_service)

* **add\_button**: кнопка, которая вызывает функцию **add\_service** для добавления нового сервиса.

**Компоновка интерфейса:**

python

layout = QVBoxLayout()

layout.addWidget(self.search\_label)

layout.addWidget(self.search\_input)

layout.addWidget(self.password\_table)

layout.addWidget(self.add\_button)

self.setLayout(layout)

self.load\_passwords()

* **layout**: вертикальный макет для размещения всех элементов интерфейса. Сюда добавляются элементы поиска, таблица и кнопка.

**5. Методы для работы с паролями**

**Загрузка паролей:**

python

def load\_passwords(self):

self.password\_table.setRowCount(0)

for service, data in self.password\_manager.passwords.items():

self.add\_to\_table(service, data['username'], data['password'])

* **load\_passwords**: загружает все пароли и добавляет их в таблицу.

**Добавление пароля в таблицу:**

python

Копировать код

def add\_service(self):

service, ok = QInputDialog.getText(self, 'Новый сервис', 'Введите название сервиса:')

if ok and service:

username, ok2 = QInputDialog.getText(self, 'Логин', 'Введите логин:')

if ok2 and username:

password, ok3 = QInputDialog.getText(self, 'Пароль', 'Введите пароль (или оставьте пустым для генерации):')

if ok3:

if not password:

password = generate\_password()

self.password\_manager.add\_password(service, username, password)

self.add\_to\_table(service, username, password)

* **add\_service**: позволяет пользователю добавить новый сервис, логин и пароль. Если пользователь не вводит пароль, генерируется случайный пароль.

**Добавление данных в таблицу:**

python

def add\_to\_table(self, service, username, password):

row = self.password\_table.rowCount()

self.password\_table.insertRow(row)

self.password\_table.setItem(row, 0, QTableWidgetItem(service))

self.password\_table.setItem(row, 1, QTableWidgetItem(username))

self.password\_table.setItem(row, 2, QTableWidgetItem(password))

* **add\_to\_table**: добавляет данные о сервисе, логине и пароле в таблицу.

**Поиск по сервису:**

python

def search\_service(self):

search\_term = self.search\_input.text().lower()

filtered\_data = {s: u for s, u in self.password\_manager.passwords.items() if search\_term in s.lower()}

self.update\_table(filtered\_data)

* **search\_service**: фильтрует сервисы по введенному тексту.

**Обновление таблицы:**

python

def update\_table(self, data):

self.password\_table.setRowCount(0)

for service, info in data.items():

self.add\_to\_table(service, info["username"], info["password"])

* **update\_table**: обновляет таблицу с результатами поиска.

**6. Класс окна для входа (LoginWindow)**

python

class LoginWindow(QWidget):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.init\_ui()

* **LoginWindow**: класс для окна входа, где пользователь вводит мастер-пароль для доступа к приложению.

**Интерфейс окна входа:**

python

def init\_ui(self):

self.setWindowTitle("Вход в систему")

self.setGeometry(100, 100, 300, 200)

self.password\_label = QLabel("Введите мастер-пароль:")

self.password\_input = QLineEdit(self)

self.password\_input.setEchoMode(QLineEdit.Password)

self.password\_input.setPlaceholderText("Введите мастер-пароль")

self.login\_button = QPushButton("Войти")

self.login\_button.clicked.connect(self.check\_password)

layout = QVBoxLayout()

layout.addWidget(self.password\_label)

layout.addWidget(self.password\_input)

layout.addWidget(self.login\_button)

self.setLayout(layout)

* **password\_input**: поле для ввода мастер-пароля.
* **login\_button**: кнопка для входа в приложение.

**Проверка пароля:**

python

def check\_password(self):

entered\_password = self.password\_input.text()

if entered\_password == "master123": # Мастер-пароль можно заменить на более сложный

self.open\_main\_app(entered\_password)

else:

QMessageBox.warning(self, "Ошибка", "Неверный мастер-пароль!")

* **check\_password**: проверяет введенный мастер-пароль и открывает главное приложение, если пароль правильный.

**7. Запуск приложения**

python

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

login\_window = LoginWindow()

login\_window.show()

sys.exit(app.exec\_())

* Создается приложение с окном входа, и программа запускается.