

Разработка приложения для генерации QR-кодов с возможностью визуализации, настройки и добавления логотипа с использованием Python

Малик Александр Игоревич

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной статье представлена разработка приложения для создания QR-кодов с возможностью добавления логотипов, изменения внешнего вида и настройки параметров, таких как размер и уровень коррекции ошибок. Приложение позволяет генерировать QR-коды для различных типов данных, включая ссылки, текстовые данные и vCard (контактную информацию). Особое внимание уделено удобству работы с графическим интерфейсом, который предоставляет пользователю возможность визуализации QR-кода и предварительного просмотра. Основой разработки является язык программирования Python и библиотеки qrcode и tkinter.

Ключевые слова: QR-код, генерация QR-кода, визуализация, Python, библиотека qrcode, логотип, графический интерфейс, настройки коррекции ошибок, vCard, мобильные технологии, контактная информация, tkinter.

Developing an application for generating QR codes with the ability to visualize, customize and add a logo using Python

Malik Aleksandr Igorevich

Sholom-Aleichem Priamursky State University

Student

Abstract

This article presents the development of an application for creating QR codes with the ability to add logos, change the appearance and adjust parameters such as size and error correction level. The application allows you to generate QR codes for various types of data, including links, text data, and vCard (contact information). Special attention is paid to the convenience of working with the graphical interface, which provides the user with the ability to visualize a QR code and preview it. The development is based on the Python programming language and the qrcode and tkinter libraries.

Keywords: QR code, QR code generation, visualization, Python, qr code library, logo, graphical interface, error correction settings, vCard, mobile technologies, contact information, tkinter.

1 Введение

1.1 Актуальность

QR-коды становятся неотъемлемой частью цифровой среды благодаря своей универсальности и простоте использования. Они используются в маркетинге, логистике, в электронной коммерции, а также для хранения контактной информации и перехода по ссылкам. Приложение для генерации QR-кодов с возможностью визуализации и вставки логотипа представляет собой инновационный инструмент, который может найти широкое применение в сферах бизнеса, образования и информационных технологий. Учитывая растущую популярность мобильных устройств и мобильных приложений, потребность в подобных инструментах продолжает увеличиваться.

1.2 Обзор исследований

Обзор исследований по внедрению QR-кодов в различных областях показывает их разнообразные применения и потенциал для оптимизации бизнес-процессов и повышения эффективности работы. В статье А.Р. Прозорова, Р.И. Каримова и Р.Г. Марданшина рассматривается внедрение QR-кодов на производстве для улучшения учёта и идентификации товаров. В ходе исследования авторы предложили использование Open Source библиотек для разработки программ, упрощающих обработку данных и снижая трудозатраты на управление техникой. Выводы исследования свидетельствуют о значительном улучшении работы благодаря снижению ошибок и автоматизации документооборота, что повышает эффективность работы на производственных предприятиях [1]. В статье Р.О. Насибулина, Е.А. Сафоновой, Д.М. Абдуллина и Г.А. Гареевой рассматривается использование Telegram бота для оптимизации инвентарного учёта основных средств. Бот взаимодействует с базой данных и позволяет проводить инвентаризацию с помощью QR-кодов. В результате работы был разработан инструмент для добавления, хранения и обработки данных об основных средствах. Это решение упрощает процессы учёта и мониторинга баланса, позволяя существенно сэкономить ресурсы на хранение программного обеспечения и упростить взаимодействие с системой [2]. В статье В.П. Маринина исследуется создание проблемно-ориентированного языка для шифрования данных в QR-кодах, использующего формат YAML. Этот подход позволяет улучшить безопасность передачи данных и повысить надёжность передачи информации. Язык предоставляет удобные инструменты для работы с зашифрованными сообщениями, что делает его полезным для применения в системах, где необходимо защитить передаваемые данные, используя QR-коды [3]. В исследовании Т.Х. Ле Миня рассматривается использование технологии блокчейн для повышения доверия потребителей к бренду с помощью отслеживания товаров через QR-коды. Исследование подчёркивает важность прозрачности в маркировке продукции и её влияния на восприятие качества товара покупателями. Использование блокчейна для маркировки товаров с QR-кодами позволяет повысить уровень доверия к бренду и

улучшить репутацию компании. Исследование также затрагивает влияние этих технологий на восприятие и поведение потребителей, что является значимым для повышения конкурентоспособности [4]. В статье Д. А. Алигуловича, З. И. Махмудовой и Р. Ф. Сайфуллаева рассматриваются плюсы и минусы использования Python и Tkinter для разработки приложений с графическим пользовательским интерфейсом (GUI). Авторы подробно анализируют потенциальные области применения Tkinter, акцентируя внимание на его возможностях для создания кроссплатформенных решений, что делает этот инструмент особенно привлекательным для использования [5].

1.3 Цель исследования

Целью данного исследования является разработка программного инструмента, который позволяет пользователю генерировать QR-коды с возможностью визуализации, настройки параметров и добавления логотипа. Программа должна быть универсальной и простотой в использовании, обеспечивая удобство в процессе создания QR-кодов для различных сценариев (ссылки, текст, контактные данные).

2 Материалы и методы

Для разработки приложения использовались следующие материалы и методы:

Язык программирования: Python 3.12.

Библиотеки:

- **qrcode** — для генерации и настройки QR-кодов.
- **tkinter** — для создания графического интерфейса пользователя (GUI).
- **Pillow** — для работы с изображениями, вставки логотипа в QR-код.
- **colorchooser** — для выбора цветов QR-кода и фона.

Методы разработки:

- Разработка графического интерфейса с использованием библиотеки tkinter.
- Создание функционала для генерации QR-кодов с настройками, включая возможность добавления логотипа.
- Реализация различных уровней коррекции ошибок для улучшения качества QR-кода.
- Визуализация QR-кодов с предпросмотром и автоматическое сохранение с возможностью выбора формата.

3 Результаты и обсуждения

1. Импорт библиотек (рис.1).

```
import qrcode
import os
from tkinter import *
from tkinter import filedialog, messagebox, colorchooser
from tkinter.ttk import Notebook
from PIL import Image, ImageTk
```

Рисунок 1. Импорт библиотек

- qrcode: библиотека для создания QR-кодов.
- os: библиотека для работы с операционной системой (например, для сохранения файла).
- tkinter: основная библиотека для создания графического интерфейса.
- PIL (Python Imaging Library): для работы с изображениями (загрузка и манипуляция с логотипами).

2. Глобальная переменная для логотипа (рис.2).

```
# Инициализация глобальной переменной для логотипа
logo_path = "" # Начальное значение для logo_path
```

Рисунок 2. Глобальная переменная

Здесь создается переменная `logo_path`, которая будет хранить путь к изображению логотипа.

3. Функции управления полями ввода для vCard (рис.3).

```
def toggle_vcard_fields(show): 4 usages
    if show:
        label_name.pack(pady=5)
        entry_name.pack(pady=5)
        label_phone.pack(pady=5)
        entry_phone.pack(pady=5)
        label_email.pack(pady=5)
        entry_email.pack(pady=5)
        label_data.pack_forget() # Скрываем поле ввода для других типов данных
        entry_data.pack_forget() # Скрываем поле ввода для других типов данных
    else:
        label_name.pack_forget() # Скрываем поля vCard
        entry_name.pack_forget()
        label_phone.pack_forget()
        entry_phone.pack_forget()
        label_email.pack_forget()
        entry_email.pack_forget()
        label_data.pack(pady=5) # Показываем поле для ввода других данных
        entry_data.pack(pady=5) # Показываем поле для ввода других данных
```

Рисунок 3. Код функции управления полями ввода для vCard

Функция `toggle_vcard_fields` управляет отображением полей ввода для создания vCard (контактной информации). Если выбран тип данных "vCard", то поля для имени, телефона и email показываются. Иначе скрываются.

4. Функция генерации QR-кода с логотипом (рис.4).

```
def generate_qrcode_with_logo(): 2 usages
    global logo_path # Указываем, что используем глобальную переменную logo_path

    data = entry_data.get()
    if not data and qr_type_var.get() != "Контакты (vCard)":
        messagebox.showwarning(title="Внимание", message="Введите данные для QR-кода!")
    return
```

Рисунок 4. Код функции генерации QR-кода

Начинаем генерацию QR-кода с проверки введенных данных. Если нет данных, и выбран не тип "vCard", выводим предупреждение.

```

if qr_type_var.get() == "Ссылка":
    data = f"http://{data}" if not data.startswith("http") else data
elif qr_type_var.get() == "Текст":
    data = data
elif qr_type_var.get() == "Контакты (vCard)":
    # Получаем данные для vCard
    name = entry_name.get()
    phone = entry_phone.get()
    email = entry_email.get()

    # Проверяем, чтобы все поля были заполнены
    if not name or not phone or not email:
        messagebox.showwarning( title: "Внимание", message: "Заполните все поля для vCard!")
        return

    # Формируем строку vCard
    data = f"""
BEGIN:VCARD
VERSION:3.0
FN:{name}
TEL:{phone}
EMAIL:{email}
END:VCARD
"""

```

Рисунок 5. Код создания строки ввода для определенного типа данных

В зависимости от типа данных (ссылка, текст или vCard) создаём соответствующую строку для QR-кода (рис.5).

```

qr = qrcode.QRCode(
    version=qr_version.get(),
    error_correction=correction_levels[correction_level],
    box_size=box_size.get(),
    border=border_size.get(),
)
qr.add_data(data)
qr.make(fit=True)

```

Рисунок 6. Код создания QR-кода с заданными настройками

Настройка и создание QR-кода с использованием выбранных параметров: версия, уровень коррекции ошибок, размер клеток и границы (рис.6).

```

img_qr = qr.make_image(fill=color_fg, back_color=color_bg)

```

Рисунок 7. Генерация qr-кода с заданным цветом

Генерация изображения QR-кода с заданным цветом (рис.7).

```

if logo_path:
    logo = Image.open(logo_path)
    qr_width, qr_height = img_qr.size
    logo_width, logo_height = logo.size

    # Масштабируем логотип
    logo_size = int(
        logo_size_var.get() * qr_width / 100) # Логотип будет занимать определенную процентную часть QR-кода
    logo = logo.resize((logo_size, logo_size))

    # Вычисляем позицию для размещения логотипа в центре
    logo_position = ((qr_width - logo_size) // 2, (qr_height - logo_size) // 2)

    # Вставляем логотип в центр QR-кода
    img_qr.paste(logo, logo_position, logo.convert("RGBA"))

```

Рисунок 8. Загрузка логотипа

Если путь к логотипу существует, загружаем его, изменяем размер и вставляем в центр QR-кода (рис.8).

```

preview_qr_code(img_qr)

# Сохранение изображения в папку с программой
file_path = os.path.join(os.getcwd(), "generated_qrcode.png")
img_qr.save(file_path)
messagebox.showinfo(title="Успех", message=f"QR-код сохранён по пути {file_path}")

```

Рисунок 9. Отображение и сохранение qr-кода

Отображаем предварительный просмотр QR-кода и сохраняем его в файл (рис.9).

5. Предпросмотр QR-кода (рис.10).

```

def preview_qr_code(img_qr): 1 usage
    img_qr.thumbnail((200, 200)) # Ограничиваем размер для предпросмотра
    img_qr_preview = ImageTk.PhotoImage(img_qr)

    qr_preview_label.config(image=img_qr_preview)
    qr_preview_label.image = img_qr_preview

```

Рисунок 10. Функция предпросмотра qr-кода

Функция для предварительного просмотра изображения QR-кода в уменьшенном размере.

6. Загрузка и удаление логотипа (рис.11.1-11.2).

```
def load_logo(): 1 usage
    global logo_path # Объявляем, что используем глобальную переменную logo_path
    logo_path = filedialog.askopenfilename(filetypes=[("Image Files", "*.png;*.jpg;*.jpeg;*.bmp;*.gif")])
    if logo_path:
        logo_image = Image.open(logo_path)
        logo_image.thumbnail((100, 100)) # Ограничиваем размер превью логотипа
        logo_image = ImageTk.PhotoImage(logo_image)

        logo_preview.config(image=logo_image)
        logo_preview.image = logo_image
        logo_label.config(text="Логотип загружен")
```

Рисунок 11.1. Функция загрузки логотипа

Загрузка логотипа из файла и отображение его в интерфейсе.

```
def remove_logo(): 1 usage
    global logo_path # Указываем, что это глобальная переменная
    logo_path = "" # Очищаем путь к логотипу
    logo_preview.config(image=None)
    logo_label.config(text="Логотип удалён, выберите новый логотип (необязательно)")
    # Регенерация QR-кода без логотипа
    generate_qrcode_with_logo()
```

Рисунок 11.2. Функция удаления логотипа

Удаление логотипа и регенерация QR-кода без логотипа.

7. Очистка ввода и регенерация QR-кода (рис.12).

```
def clear_input(): 1 usage
    entry_data.delete(first=0, END)
    entry_name.delete(first=0, END)
    entry_phone.delete(first=0, END)
    entry_email.delete(first=0, END)
    qr_type_var.set("Ссылка")
    logo_preview.config(image=None)
    logo_label.config(text="Выберите логотип (необязательно)")
    qr_preview_label.config(image=None)
    global logo_path
    logo_path = ""
```

Рисунок 12. Функция очистки ввода и регенерация qr-кода

Очистка всех полей ввода и сброс всех настроек.

8. Выбор цвета для QR-кода и фона (рис.13).

```
# Функция для выбора цвета QR-кода
def choose_color_fg(): 1 usage
    global color_fg
    color_fg = colorchooser.askcolor(title="Выберите цвет QR-кода")[1]

# Функция для выбора цвета фона
def choose_color_bg(): 1 usage
    global color_bg
    color_bg = colorchooser.askcolor(title="Выберите цвет фона")[1]
```

Рисунок 13. Функция выбора цвета qr-кода и фона

Выбор цветов для QR-кода и фона через диалоговое окно.

9. Главное окно и интерфейс.

Создание главного окна и настройка интерфейса, включая вкладки, радиокнопки для выбора типа данных, поля ввода, кнопки и другие элементы управления (рис.14.1-14.3).

```

# Создание главного окна приложения
root = Tk()
root.title("Усложнённое приложение для создания QR-кодов с логотипом")

# Настройка вкладок
tab_control = Notebook(root) # Теперь этот класс доступен, так как мы импортировали его
tab_input = Frame(tab_control)
tab_settings = Frame(tab_control)

tab_control.add(tab_input, text="Ввод данных")
tab_control.add(tab_settings, text="Настройки")

tab_control.pack(expand=1, fill="both")

# Вкладка 1: Ввод данных
qr_type_var = StringVar(value="Ссылка")

radio_link = Radiobutton(tab_input, text="Ссылка", variable=qr_type_var, value="Ссылка",
                        command=lambda: toggle_vcard_fields(False))
radio_link.pack(anchor=W)
radio_text = Radiobutton(tab_input, text="Текст", variable=qr_type_var, value="Текст",
                        command=lambda: toggle_vcard_fields(False))
radio_text.pack(anchor=W)
radio_vcard = Radiobutton(tab_input, text="Контакты (vCard)", variable=qr_type_var, value="Контакты (vCard)",
                        command=lambda: toggle_vcard_fields(True))
radio_vcard.pack(anchor=W)

# Ввод данных для vCard
label_name = Label(tab_input, text="Имя:")
label_name.pack(pady=5)
entry_name = Entry(tab_input, width=50)
entry_name.pack(pady=5)

label_phone = Label(tab_input, text="Телефон:")
label_phone.pack(pady=5)
entry_phone = Entry(tab_input, width=50)
entry_phone.pack(pady=5)

label_email = Label(tab_input, text="Email:")
label_email.pack(pady=5)
entry_email = Entry(tab_input, width=50)
entry_email.pack(pady=5)

# Ввод данных для других типов QR-кодов
label_data = Label(tab_input, text="Введите данные для QR-кода:")
label_data.pack(pady=5)

entry_data = Entry(tab_input, width=50)
entry_data.pack(pady=5)

# Начально скрыть поля ввода для vCard
toggle_vcard_fields(False)

```

Рисунок 14.1. Код

```

logo_button = Button(tab_input, text="Загрузить логотип", command=load_logo)
logo_button.pack(pady=10)

logo_preview = Label(tab_input)
logo_preview.pack(pady=10)

logo_label = Label(tab_input, text="Выберите логотип (необязательно)")
logo_label.pack(pady=5)

remove_logo_button = Button(tab_input, text="Удалить логотип", command=remove_logo)
remove_logo_button.pack(pady=5)

button_generate = Button(tab_input, text="Создать QR-код с логотипом", command=generate_qrcode_with_logo)
button_generate.pack(pady=10)

button_clear = Button(tab_input, text="Очистить", command=clear_input)
button_clear.pack(pady=5)

# Вкладка 2: Настройки
# Настройки размеров и уровня коррекции
label_size = Label(tab_settings, text="Размер QR-кода:")
label_size.pack(pady=5)
box_size = Scale(tab_settings, from_=5, to=20, orient=HORIZONTAL)
box_size.set(10)
box_size.pack()

label_version = Label(tab_settings, text="Версия QR-кода (1-40):")
label_version.pack(pady=5)
qr_version = Scale(tab_settings, from_=1, to=40, orient=HORIZONTAL)
qr_version.set(1)
qr_version.pack()

label_border = Label(tab_settings, text="Размер границы:")
label_border.pack(pady=5)
border_size = Scale(tab_settings, from_=1, to=10, orient=HORIZONTAL)
border_size.set(4)
border_size.pack()

label_logo_size = Label(tab_settings, text="Размер логотипа (% от QR-кода):")
label_logo_size.pack(pady=5)
logo_size_var = Scale(tab_settings, from_=5, to=30, orient=HORIZONTAL)
logo_size_var.set(10)
logo_size_var.pack()

# Добавление выпадающего списка для уровня коррекции
correction_level_var = StringVar(value='L')
label_correction = Label(tab_settings, text="Уровень коррекции ошибок:")
label_correction.pack(pady=5)
correction_level_menu = OptionMenu(tab_settings, correction_level_var, value='L', *values='M', 'Q', 'H')
correction_level_menu.pack()

# Цвета для QR-кода и фона
color_fg = "black"
color_bg = "white"

```

Рисунок 14.2. Код

```

color_fg_button = Button(tab_settings, text="Выберите цвет QR-кода", command=choose_color_fg)
color_fg_button.pack(pady=10)

color_bg_button = Button(tab_settings, text="Выберите цвет фона", command=choose_color_bg)
color_bg_button.pack(pady=10)

# Метка для предпросмотра QR-кода
qr_preview_label = Label(tab_input)
qr_preview_label.pack(pady=10)

root.mainloop()

```

Рисунок 14.3. Код

10. Внешний вид программы и пример qr-кода (рис.15).

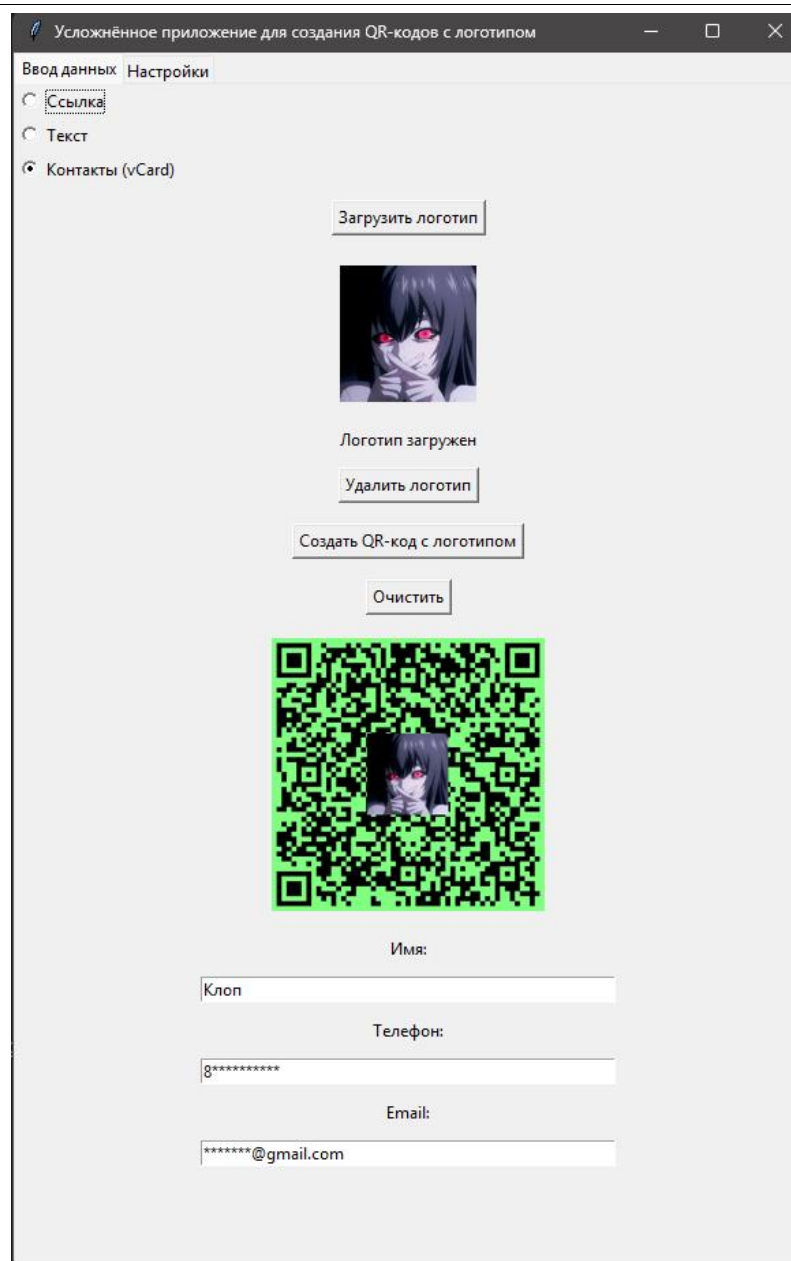


Рисунок 15. Внешний вид и пример qr-кода

Выводы

Разработанное приложение является удобным инструментом для создания QR-кодов, позволяющим адаптировать их под разные типы данных и настраивать внешний вид в зависимости от потребностей пользователя. Возможность добавления логотипа и выбора цветовых схем делает приложение подходящим для применения в различных областях, таких как маркетинг, бизнес и персональное использование. Приложение может быть полезным для малых и средних предприятий, а также для индивидуальных пользователей, которые ищут простое и эффективное решение для создания и использования QR-кодов в различных целях. В дальнейшем возможно расширение функционала, включая поддержку дополнительных типов данных и интеграцию с другими системами для автоматизации процессов.

Библиографический список

1. Прозоров А.Р., Каримов Р.И., Марданшин Р.Г. Внедрение qr-кодов на производстве // Новые технологии наукоемкого машиностроения. 2019. С. 204-207.
2. Насибулин Р.О., Сафонова Е.А., Абдуллин Д.М., Гареева Г.А. Оптимизация инвентарного учета основных средств путем создания telegram бота // International journal of advanced studies. 2023. Т. 13. № 1. С. 90-101.
3. Маринин В.П. Проблемно-ориентированный язык описания генерации qr-кодов // Кибербезопасность: технические и правовые аспекты защиты информации. 2022. С. 181-184.
4. Minh T.H.L. Fostering product quality and Brand Trust by QR code traceability and customer reviews: The moderating role of brand reputation in Blockchain // The journal of high technology management research. 2024. Т.35. №. 1.
5. Aligulovich D.A., Mahmudova Z.I., Sayfullayev R.F. Tibbiyot yo'nalishida zamonaviy pedagogik texnologiyalarini qo'llab tkinter paketida gui dasturlarini tuzish // Евразийский журнал математической теории и компьютерных наук. 2022. №2(12). С. 31-35.