# Dokumentacja techniczna – "Ruletka Kasyno"

## Opis projektu

**"Ruletka Kasyno"** to wieloplatformowa gra stworzona w języku Python, która symuluje klasyczną ruletkę w stylu kasynowym. Projekt wykorzystuje bibliotekę **Pygame** do tworzenia interfejsu graficznego, **Tkinter** do obsługi formularzy logowania i rejestracji, oraz **SQLite** jako lokalną bazę danych użytkowników.

Gra pozwala na:

- tworzenie i rejestrację graczy,
- logowanie się z użyciem hasła,
- zarządzanie saldem konta,
- wpisywanie zakładów,
- kręcenie ruletką i wygrywanie nagród.

Interfejs jest prosty i przejrzysty, a całość działa w trybie lokalnym – bez potrzeby połączenia z internetem.

# Główne menu gry (menu.py)

**1. Opis ogólny**

Plik ten realizuje główne menu gry **„Ruletka Kasyno”**, stworzonej w języku **Python** z użyciem biblioteki **Pygame**.
Po uruchomieniu aplikacji użytkownik widzi ekran powitalny z tytułem gry oraz przyciskiem **START**, który przenosi do właściwej części rozgrywki. Menu obsługuje również zamknięcie aplikacji.

**2. Zależności**

python

KopiujEdytuj

import pygame

import sys

from player_screen import main_loop

from db import init_db

- pygame – główna biblioteka do obsługi grafiki, zdarzeń i okna gry.

- sys – używana do zamknięcia aplikacji (sys.exit()).
- player_screen.main_loop – funkcja uruchamiająca rozgrywkę.
- db.init_db – funkcja inicjalizująca bazę danych.

**3. Inicjalizacja**

python

KopiujEdytuj

init_db()

pygame.init()

- init_db() – przygotowuje środowisko bazodanowe (np. tworzy tabele, otwiera połączenia).
- pygame.init() – inicjalizuje wszystkie potrzebne moduły biblioteki Pygame.

**4. Konfiguracja okna gry**

python

KopiujEdytuj

WIDTH, HEIGHT = 1000, 700

screen = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption("Ruletka Kasyno - Menu")

- Ustawienie rozdzielczości okna gry na **1000x700 pikseli**.
- Nadanie tytułu okna: **"Ruletka Kasyno - Menu"**.

**5. Ustawienia czcionek i kolorów**

python

KopiujEdytuj

font = pygame.font.SysFont(None, 48)

WHITE = (255, 255, 255)

BLUE = (0, 0, 255)

GREEN = (0, 100, 0)

- Czcionka systemowa o rozmiarze **48 punktów**.
- Definicje kolorów w formacie RGB.

**6. Pętla główna menu**

python

KopiujEdytuj

running = True

while running:

...

- Pętla działa dopóki użytkownik nie zamknie aplikacji (pygame.QUIT).

**7. Renderowanie ekranu**

**7.1. Tło**

python

KopiujEdytuj

screen.fill(GREEN)

- Ustawienie koloru tła na zielony (symbolika stołu kasynowego).

**7.2. Tytuł gry**

python

KopiujEdytuj

title_text = font.render("Witamy w Ruletce Kasyno", True, WHITE)

screen.blit(title_text, (srodek_x, srodek_y - 100))

- Renderowanie napisu tytułowego i jego wyśrodkowanie.

**7.3. Przycisk START**

python

KopiujEdytuj

start_btn = pygame.Rect(WIDTH // 2 - 100, HEIGHT // 2 - 30, 200, 60)

pygame.draw.rect(screen, BLUE, start_btn)

start_text = font.render("START", True, WHITE)

screen.blit(start_text, (WIDTH // 2 - start_text.get_width() // 2, HEIGHT // 2 - 15))

- Rysowanie niebieskiego prostokąta i umieszczenie na nim tekstu „START”.

**8. Obsługa zdarzeń**

python

KopiujEdytuj

```
for event in pygame.event.get():
   if event.type == pygame.QUIT:
      running = False
   elif event.type == pygame.MOUSEBUTTONDOWN:
      if start_btn.collidepoint(event.pos):
         main_loop()
```

- Jeśli użytkownik kliknie „X” – gra zostaje zamknięta.
- Jeśli kliknie przycisk „START” – zostaje uruchomiona główna rozgrywka (main_loop()).

**9. Odświeżanie ekranu**

python

KopiujEdytuj

```
pygame.display.flip()
```

- Aktualizacja zawartości wyświetlacza – pokazanie wszystkich zmian.

**10. Zakończenie programu**

python

KopiujEdytuj

```
pygame.quit()
sys.exit()
```

- Zwolnienie zasobów zajmowanych przez Pygame.
- Zamknięcie aplikacji w sposób bezpieczny.

# Ekran graczy i zarządzanie kontami (player_screen.py)

**1. Opis ogólny**

Ten moduł odpowiada za **interfejs zarządzania graczami** w grze „Ruletka Kasyno”. Umożliwia:

- Dodawanie graczy poprzez rejestrację lub logowanie,

- Wyświetlanie listy graczy z ich saldami,
- Usuwanie graczy,
- Dodawanie pieniędzy (instrukcje przelewu),
- Rozpoczęcie gry w ruletkę (jeśli są gracze z wystarczającym saldem).

Moduł wykorzystuje biblioteki pygame (interfejs graficzny) oraz tkinter (okna dialogowe), a także funkcje z plików db.py (rejestracja, logowanie) i roulette.py (uruchomienie gry).

**2. Zależności**

python

KopiujEdytuj

import pygame

import tkinter as tk

import threading

import re

from tkinter import messagebox

from db import register_user, authenticate_user

import sys

from roulette import launch_roulette

**Opis bibliotek:**

- pygame – główny silnik graficzny gry.
- tkinter – tworzenie okienek GUI do logowania, rejestracji itp.
- threading – uruchamianie okien tkinter w osobnych wątkach (bez blokowania Pygame).
- re – walidacja danych (np. PESEL, nick).
- messagebox – komunikaty błędów i informacji.
- db – obsługa użytkowników: register_user(), authenticate_user().
- roulette – uruchomienie gry ruletki z przekazanymi graczami.

**3. Ustawienia początkowe**

python

KopiujEdytuj

pygame.init()

WIDTH, HEIGHT = 1000, 700

FONT = pygame.font.SysFont("arial", 24, bold=True)

- Inicjalizacja silnika gry i ustawienie rozdzielczości.
- Zdefiniowanie czcionki oraz kolorów interfejsu (WHITE, BLACK, GREEN, RED, itp.).

Tworzone jest także okno:

python

KopiujEdytuj

screen = pygame.display.set_mode((WIDTH, HEIGHT))

pygame.display.set_caption("Ruletka Kasyno - Menu i Gracze")

**4. Struktura danych**

python

KopiujEdytuj

players = []

delete_buttons = []

- players: lista słowników, gdzie każdy gracz ma nick i balance.
- delete_buttons: lista prostokątów (pygame.Rect) przypisanych do graczy.

**5. Funkcja draw_ui()**

Wyświetla cały interfejs graficzny:

- Przycisk "Wróć"
- Lista graczy i ich sald
- Przycisk "usuń" przy każdym graczu
- Przycisk "+ Dodaj nowego użytkownika"
- Przycisk "Dodaj pieniędzy" (otwiera instrukcje przelewu)
- Przycisk "Gramy!" – przechodzi do gry

Jeśli graczy jest mniej niż 4 – pokazuje opcję dodania nowego.

**6. Funkcja show_add_funds_message()**

Otwiera nowe okno z instrukcjami przelewu. Pokazuje dane konta bankowego i tytuł przelewu (nick gracza).

**7. Funkcja open_user_action_window()**

Okno wyboru akcji:

- „Zaloguj się” – uruchamia open_login_window()
- „Zarejestruj się” – uruchamia open_registration_window()

**8. Funkcja open_login_window()**

Formularz logowania użytkownika:

- Sprawdza, czy nick nie jest już na liście.
- Weryfikuje dane przez authenticate_user().
- Po poprawnym logowaniu – dodaje gracza do listy.

**9. Funkcja open_registration_window()**

Formularz rejestracji nowego gracza:

- Zbiera dane: nick, imię, nazwisko, PESEL, hasło.
- Waliduje dane (np. PESEL = 11 cyfr, bez cyfr w imieniu).
- Jeśli rejestracja przez register_user() powiedzie się – dodaje gracza do listy z saldem 2000.

**10. Funkcja main_loop()**

Główna pętla dla tego modułu:

- Obsługuje kliknięcia użytkownika:
    - Przycisk „Dodaj nowego użytkownika” – otwiera okno wyboru akcji.
    - Przycisk „Dodaj pieniędzy” – otwiera instrukcję przelewu.
    - Przycisk „Wróć” – kończy pętlę i wraca do wcześniejszego widoku.
    - Przycisk „Gramy!” – usuwa graczy z saldem <25, następnie uruchamia launch_roulette(players).
- Dla każdego kliknięcia „usuń” – usuwa odpowiedniego gracza z listy.

**11. Walidacje danych**

W rejestracji i logowaniu sprawdzane są m.in.:

- Poprawność nicku (długość, znaki),

- Imię/nazwisko bez cyfr,
- PESEL: dokładnie 11 cyfr,
- Spójność haseł i nicków w bazie danych.

**12. Warunki rozpoczęcia gry**

Gra może się rozpocząć po kliknięciu „Gramy!”, jeśli:

- Jest co najmniej jeden gracz,
- Wszyscy gracze mają min. 25 jednostek pieniędzy.

Gracze z saldem poniżej 25 są usuwani, a użytkownik otrzymuje komunikat.

**13. Zakończenie**

Pętla main_loop() działa dopóki nie zostanie zamknięta przez:

- kliknięcie „Wróć”,
- zamknięcie okna gry (pygame.QUIT).

Po zakończeniu main_loop() gra może wrócić do menu głównego lub innego etapu.

# Moduł gry „Ruletka” (roulette.py)

**1. Opis ogólny**

Ten moduł obsługuje **interaktywną rozgrywkę w ruletkę** w ramach gry „Ruletka Kasyno”. Zawiera:

- graficzne przedstawienie koła ruletki i strzałki,
- pola tekstowe dla graczy do wpisania zakładów (numerów),
- rozwijane menu wyboru stawki,
- logikę losowania wyniku i aktualizacji sald graczy.

Zbudowany jest na bazie pygame, pygame_gui oraz tkinter (do wyświetlania komunikatów).

**2. Zależności**

python

KopiujEdytuj

```
import pygame
import pygame_gui
import tkinter as tk
from tkinter import messagebox
import random, time, sys
from db import update_balance
```

**Wykorzystane biblioteki:**

- pygame – do tworzenia okna gry, renderowania ruletki, obsługi zdarzeń.
- pygame_gui – obsługa formularzy tekstowych i rozwijanego menu zakładów.
- tkinter.messagebox – okna z komunikatami (np. wygrane, błędy).
- random – losowanie wyników ruletki.
- time – zarządzanie opóźnieniami i animacjami.
- db.update_balance – aktualizacja stanu konta gracza w bazie danych.

**3. Funkcja launch_roulette(players)**

**Parametr:**

- players: lista graczy, każdy jako słownik z kluczami nick, balance.

**Działanie:**

- Inicjalizuje okno gry i GUI.
- Tworzy pola tekstowe dla każdego gracza do wpisania zakładu.
- Pozwala uruchomić losowanie i aktualizuje wyniki.
- Wyświetla zwycięzców i zmiany w saldach.

**4. Funkcje pomocnicze**

**show_message(title, message)**

- Tworzy wyskakujące okno z komunikatem (tkinter.messagebox).

**setup_inputs(players, manager)**

- Dla każdego gracza tworzy pole tekstowe do wpisania obstawionego numeru.
- Zwraca listę słowników {player, input}.

**validate_inputs(inputs, numbers, selected_bet)**

- Sprawdza poprawność każdego zakładu:
    - saldo ≥ 25 zł,
    - saldo ≥ wybrany zakład,
    - wpisany numer istnieje w ruletce.
- Zwraca (True, "") lub (False, komunikat).

**start_spin(current_angle, result_number, get_angle_for_number)**

- Oblicza cel obrotu ruletki tak, aby zatrzymała się na wybranym numerze.
- Dodaje kilka pełnych obrotów dla efektu.

**handle_spin_result(inputs, result_number, bet_amount)**

- Porównuje wpisane numery graczy z wylosowanym.
- Nagroda za trafienie = stawka * 36.
- Przegrani tracą postawioną kwotę.
- Aktualizuje bazę danych przez update_balance.

**5. Interfejs graficzny**

**Koło ruletki**

- Obraz ruletka1.png – skalowany do odpowiedniego rozmiaru.
- Obraca się w zależności od wylosowanego wyniku.

**Strzałka**

- Obraz strzalka.png wskazuje miejsce zatrzymania ruletki.

**Zakłady**

- Gracze wpisują numer w polu tekstowym.
- Zakład wybierany z rozwijanego menu (UIDropDownMenu): 25, 50, 100, 500.

**6. Główna pętla gry (while running)**

Obsługuje:

- Rysowanie interfejsu (draw_ui()),
- Kliknięcia w przycisk „Wróć” – kończy grę i wraca do poprzedniego ekranu,

- Kliknięcia w przycisk „Kręć” – walidacja zakładów i rozpoczęcie obrotu,
- Obliczanie obrotu ruletki (spin_speed, slowdown_distance),
- Wyświetlenie wyników i komunikatu kto wygrał.

**7. Logika koła ruletki**

- Lista numerów: ["0", "28", ..., "2"] – pełna sekwencja ruletki (wraz z „00”).
- Funkcja get_angle_for_number() przelicza numer na kąt.
- Obrót = pełne 7 obrotów + potrzebna różnica do zatrzymania na wyniku.

**8. Wyniki rozgrywki**

Po zatrzymaniu ruletki:

- Funkcja handle_spin_result() aktualizuje saldo graczy.
- Wyświetlany jest komunikat o wygranych (lub ich braku).
- Pola wejściowe graczy są czyszczone.

**9. Obsługa błędów**

- Brak środków na koncie (poniżej zakładu): wyświetla komunikat.
- Zły numer (nie istnieje w ruletce): komunikat o błędzie.
- Tylko poprawne dane pozwalają uruchomić „Kręć”.

**10. Warunki zakończenia**

- Kliknięcie „Wróć” kończy działanie launch_roulette.
- pygame.QUIT kończy całkowicie aplikację (sys.exit()).

**11. Wymagane pliki**

- ruletka1.png – obraz tarczy ruletki.
- strzalka.png – obraz wskaźnika.
- Własna baza danych (przez db.update_balance).

# Baza Danych (db.py)

**1. Opis ogólny**

Plik db.py obsługuje bazę danych SQLite dla gry "Ruletka Kasyno". Odpowiada za:

- Rejestrację graczy,
- Autoryzację (logowanie),
- Tworzenie tabeli danych,
- Aktualizację salda gracza po grze.

Wszystkie operacje są zabezpieczone przed równoległym dostępem za pomocą blokady (threading.Lock), co zapewnia bezpieczeństwo w środowisku wielowątkowym (np. przy jednoczesnym korzystaniu z Pygame i Tkintera).

**2. Zależności**

python

KopiujEdytuj

import sqlite3

import threading

Biblioteki:

- sqlite3 – standardowa biblioteka Pythona do obsługi baz danych SQLite,
- threading – służy do kontroli dostępu w przypadku współbieżności.

**3. Stałe i blokada**

python

KopiujEdytuj

DB_PATH = 'players.db'

db_lock = threading.Lock()

- DB_PATH – ścieżka do pliku bazy danych.
- db_lock – globalna blokada zapobiegająca równoczesnemu zapisywaniu do bazy z wielu wątków.

**4. Funkcja init_db()**

Tworzy bazę danych i tabelę players, jeśli nie istnieje.

python

KopiujEdytuj

```
def init_db():
    with sqlite3.connect(DB_PATH) as conn:
        cursor = conn.cursor()
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS players (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                nick TEXT NOT NULL UNIQUE,
                first_name TEXT NOT NULL,
                last_name TEXT NOT NULL,
                pesel TEXT NOT NULL UNIQUE,
                password TEXT NOT NULL,
                balance INTEGER DEFAULT 2000
            )
        ''')
        conn.commit()
```

Kolumny tabeli:

- id – klucz główny (autonumerowany),
- nick – unikalny pseudonim gracza,
- first_name i last_name – imię i nazwisko,
- pesel – unikalny numer PESEL,
- password – hasło (przechowywane jawnie),
- balance – stan konta (domyślnie 2000).

**5. Funkcja register_user(...)**

Rejestruje nowego użytkownika w bazie danych.

python

KopiujEdytuj

```
def register_user(nick, first_name, last_name, pesel, password):
    with db_lock:
```

```
    try:
        with sqlite3.connect(DB_PATH) as conn:
            cursor = conn.cursor()
            cursor.execute(
                "INSERT INTO players (nick, first_name, last_name, pesel, password) VALUES (?, ?, ?, ?,
?)",
                (nick, first_name, last_name, pesel, password)
            )
            conn.commit()
            return True, None
    except sqlite3.IntegrityError as e:
        if 'nick' in str(e):
            return False, "Nick już istnieje"
        return False, "PESEL już istnieje"
```

Zwraca:

- (True, None) – gdy rejestracja się powiodła,
- (False, "Nick już istnieje") – gdy nick jest już w bazie,
- (False, "PESEL już istnieje") – gdy pesel już istnieje.

**6. Funkcja authenticate_user(...)**

Sprawdza, czy użytkownik istnieje i podał poprawne hasło.

python

KopiujEdytuj

```
def authenticate_user(nick, password):
  with db_lock:
    with sqlite3.connect(DB_PATH) as conn:
        cursor = conn.cursor()
        cursor.execute(
            "SELECT nick, balance FROM players WHERE nick=? AND password=?",
            (nick, password)
        )
```

```
        result = cursor.fetchone()
        return result if result else None
```

Zwraca:

- (nick, balance) – jeśli dane logowania są poprawne,
- None – jeśli login lub hasło są błędne.

**7. Funkcja update_balance(...)**

Aktualizuje saldo gracza po rozegranej rundzie.

python

KopiujEdytuj

```
def update_balance(nick, new_balance):
    with db_lock:
        with sqlite3.connect(DB_PATH) as conn:
            cursor = conn.cursor()
            cursor.execute("UPDATE players SET balance = ? WHERE nick = ?", (new_balance, nick))
            conn.commit()
```

Parametry:

- nick – identyfikator gracza,
- new_balance – nowa wartość salda (int).

**8. Wątki i bezpieczeństwo**

Wszystkie operacje na bazie są otoczone przez with db_lock:. Gwarantuje to:

- bezpieczeństwo przy jednoczesnym dostępie z różnych interfejsów (np. Pygame + Tkinter),
- unikanie konfliktów i błędów zapisu przy operacjach wielowątkowych.

**9. Ograniczenia**

- Hasła są przechowywane w postaci jawnej. Rekomendowane jest zastosowanie haszowania (np. z hashlib.sha256).
- Brak obsługi logowania błędów.
- Nie ma funkcji usuwania lub edytowania danych gracza.

# Podsumowanie

Projekt "Ruletka Kasyno" to funkcjonalna gra w ruletkę dla wielu graczy, oferująca:

- pełne zarządzanie użytkownikami,
- system rejestracji i logowania z walidacją danych,
- dynamiczną grafikę ruletki z realistyczną animacją,
- prostą, bezpieczną bazę danych z aktualizacją sald.

Całość została zaprojektowana w sposób modularny i łatwy do rozbudowy – np. o nowe typy zakładów, ranking graczy, historię rozgrywek czy sieciową rozgrywkę online.

Projekt może służyć jako baza edukacyjna do nauki Pythona, GUI, baz danych i programowania gier.