Symulacja katalogu bibliotecznego i biblioteki

1. Wykorzystane technologie
    - Język programowania: Python 3.8
        - Time – wczytanie czasu rzeczywistego do obliczeń na czasie
        - Random – generowanie liczb pseudolosowych, losowanie z listy
        - Math – obliczenia matematyczne, np. zaokrąglanie
        - Pymysql - łączenie z bazą danych
        - Numpy - działania na macierzach (wymóg biblioteki Matplotlib)
        - Matplotlib – tworzenie wykresów na podstawie danych
        - Openpyxl – odczytywanie danych z pliku .xlsx
    - Serwer bazy danych: XAMPP
        - Apache – Serwer php do obsługi phpMyAdmin
        - MySQL – Baza danych

2. Założenia symulacji
    - 1 sekunda czasu rzeczywistego to 1 minuta symulacji.
    - Biblioteka przyjmuje klientów przez 8 godzin jednak jeśli jacyś klienci są w kolejce to muszą zostać obsłużeni.
    - Klient jest zadowolony jeśli stoi w kolejce mniej lub równo 5 minut.
    - Wszystkie symulowane dni są dniami roboczymi

3. Podstawowe zmienne tworzone na początku działania programu

```
11  cnxn = pymysql.connect(host="localhost", user="root", passwd="", database="Library")
12  cursor = cnxn.cursor()
13  cursor.execute("SELECT MAX(Id) as MaxID from books")
14  maxId = cursor.fetchone()[0]
```

cnxn – utworzenie połączenia z bazą danych

cursor – obiekt kursora

cursor.execute("SELECT MAX(Id) as MaxID from books") *maxId = cursor.fetchone()[0]*	Przypisanie zmiennej *maxId* wartości maksymalnego *ID* z tabeli *books*

4. Wgrywanie książek do bazy danych z arkusza Excel (pliku .xlsx)
    a) Listę książek pobraliśmy ze strony: http://dolnoslaski.pzn.org.pl/wp-content/uploads/2014/04/spis_ksiazek_w_formacie_czytak.xls oraz zmodyfikowaliśmy plik do potrzeb projektu zostawiając tylko arkusz: Autor.
    b) Struktura pliku books.xlsx:

1	Tytuł	Autor	Sygn.
2	Młyn w piekarni	Abramow Newerly Jarosław	5089
3	Lwy wyzwolone	Abramow Newerly Jarosław	5194
4	Lwy STS-u	Abramow Newerly Jarosław	5462
5	Azazel	Akunin Boris	5348
6	Kochanek śmierci	Akunin Boris	5410
7	Diamentowa karoca 1	Akunin Boris	5494
8	Diamentowa karoca 2	Akunin Boris	5495
9	Trójgraniasty kapelusz	Alarcón Pedro Antonio	20009
10	Kalahari	Albiński Wojciech	5146

c) Struktura bazy danych, tabela książki (books)

ID	Title	Author	Signature	Amount
1	Młyn w piekarni	Abramow Newerly Jarosław	5089	0
2	Lwy wyzwolone	Abramow Newerly Jarosław	5194	1
3	Lwy STS-u	Abramow Newerly Jarosław	5462	0

W odróżnieniu do tableki w pliku books.xlsx zostały dołożone dwie kolumny:

- ID – klucz podstawowy z automatyczną inkrementacją

- Amunt – Liczba danej książki w zbiorze, generowana pseudolosowo

d) Kod programu wgrywający książki do bazy danych

```
23  def ResetDatabaseBooks():
24      cursor.execute("DELETE FROM books;")
25      cnxn.commit()
26      cursor.execute("ALTER TABLE books AUTO_INCREMENT = 1")
27      cnxn.commit()
28      workbook = load_workbook(filename="books.xlsx")
29      sheet = workbook.active
30      print("----Rozpoczęto wgrywanie książek do bazy danych----")
31      for i in range(2, 816):
32          title = "'" + sheet[f"A{i}"].value + "'"
33          author = "'" + sheet[f"B{i}"].value + "'"
34          signature = "'" + str(sheet[f"C{i}"].value) + "'"
35          amount = int(random.randint(1, 5))
36          cursor.execute("INSERT INTO books(Title, Author, Signature, Amount) VALUES (%s,%s,%s,%d)" % (
37          title, author, signature, amount))
38          cnxn.commit()
39      print("----Ukończono wgrywanie książek do bazy danych----")
40
```

Wykorzystane zmienne:

title – pobiera tytuł książki oraz dodaje przed i za apostrof wymagany przy funkcji *INSERT* języka MySQL

author - pobiera autora książki oraz dodaje przed i za apostrof wymagany przy funkcji *INSERT* języka MySQL

signature - pobiera sygnaturę książki oraz dodaje przed i za apostrof wymagany przy funkcji *INSERT* języka MySQL

amount – liczba książek w zbiorze generowana pseudolosowo przez *random.randint* z zakresu od 1 do 5

Funkcja *cursor.execute()* wykonuje zapytanie MySQL wewnątrz programu napisanego w języku Python. W naszym przypadku jest to zapytanie *INSERT INTO <nazwa tabeli(nazwy kolumn)> VALUES <wartości do wprowadzenia>*. Jednak w celu edycji zapytania w zależności od danych do wprowadzenia wykorzystaliśmy specyfikę języka Python: *%s* – wprowadzanie do zapytania zmiennej string, *%d* – wprowadzenie do zapytania zmiennej liczbowej.

Funkcja *cnxnr.commit()* zatwierdza zmiany w bazie danych.

5. Czyszczenie tabeli clients w bazie danych
    a) Struktura tabeli clients

ID	Name	Lastname	MonthDATE	DayDATE	HourDATE	MinuteDATE	ArrivalTime	OrderTime	ServiceTime	QueueTime	Pleased	BorrowedBook	ReturnTime
1	Olaf	Morski	1	1	0	2	2	1.62	1.62	0	YES	162	1

ID – klucz podstawowy z automatyczną inkrementacją
Name – losowo generowane imię klienta z listy
Lastname – losowo generowane nazwisko klienta z listy
MonthDATE – miesiąc przyjścia klienta
DayDATE – dzień przyjścia klienta
HourDATE – godzina przyjścia klienta
MinuteDATE – minuta przyjścia klienta
ArrivalTime – czas w jakim przyszedł klient liczony w minutach na dzień
OrderTime – czas spędzony przy okienku generowany rozkładem gaussa(liczba średnia – 3, odchylenie standardowe – 2)
ServiceTime – czas obsługi – suma czasu w kolejce i czasu spędzonego przy okienku
QueueTime – czas spędzony w kolejce zależny od parametrów poprzednich klientów
Pleased – zadowolenie klienta zależne od czasu spędonego w kolejce
    Pleased = YES jeśli *QueueTime < 5*
    Pleased = NO jeśli *QueueTime > 5*
BorrowedBook – ID wypożyczonej książki
ReturnTime – Czas opóźnienia w oddawaniu książki liczony w dniach:
    50% - na opóźnienie równe 0 dni
    25% - na opóźnienie równe 1 dzień
    25% - na opóźnienie równe 2 dni

    b) Kod czyszczący bazę danych przy włączeniu symulacji

```
17  def CleanDatabaseClients():
18      cursor.execute("DELETE FROM clients;")
19      cnxn.commit()
20      cursor.execute("ALTER TABLE clients AUTO_INCREMENT = 1")
21      cnxn.commit()
```

DELETE FROM clients – usunięcie wszystkich rekordów z tabeli clients
ALTER TABLE clients AUTO_INCREMET = 1 – zresetowanie automatycznej inkrementacji kolumny ID

6. Struktura klasy Client
    a) Client zawiera listę imion i nazwisk do generowania imiona i nazwiska klienta.

b) Klasa używa biblioteki random do generowania czasu obsługi.

```
def ServiceTime():
    x = random.gauss(3, 2)
    if (x < 0 or x > 10):
        return ServiceTime()
    else:
        return x
```

Czas obsługi generowany jest rozkładem normalnym o średniej arytmetycznej 3 i odchyleniu standardowym równym 2. Jeśli jest on mniejszy od 0 lub większy od 10 generowany jest on od nowa.

c) Zmienne klasy:

```
class Client:
    returnTime = 0
    queueTime = 0
    borrowedBook = 0
    def __init__(self,timeNow, day, month):
        self.serviceTime = round(ServiceTime(),2)
        self.arrivalTime = round(timeNow)
        self.name = random.choice(imiona)
        self.lastname = random.choice(nazwiska)
        self.orderTime = self.serviceTime
        self.day = day
        self.month = month
```

Klasa posiada wiele zmiennych, które pozwalają między innymi na określanie czy osoba jest jeszcze w kolejce czy też kiedy ma oddać książkę.

7. Wypożyczanie książek przez klienta
   a) Funkcja przypisująca wypożyczoną książkę do klienta

```
41  def BorrowABook(client):
42      randomBookId = random.randint(1, maxId)
43      cursor.execute(f"Select Amount FROM BOOKS WHERE Id = '{randomBookId}'")
44      quantity = cursor.fetchone()[0]
45      if (quantity != 0):
46          cursor.execute(f"UPDATE Books SET Amount = Amount - 1 Where Id ='{randomBookId}'")
47          client.borrowedBook = randomBookId
48          client.returnTime = [0,0,1,2].pop(random.randint(0,3))
49          cnxn.commit()
```

randomBookId – przypisanie do zmiennej losowego *ID* z zakresu od 1 do *maxId*

cursor.execute(f"Select Amount FROM BOOKS WHERE Id = {randomBookId}'") *quantity = cursor.fetchone()[0]*	Przypisanie zmiennej *quantity* wartości *Amount* dla danego ID

Jeśli wartość zmiennej *quantity* jest różna od 0 zostaje zmniejszona wartość *Amount* oraz przypisanie wartości do obiektu client.

8. Zwracanie książki przez klienta
    a) Funkcja zwracająca książkę do bazy danych

```
51  def ReturnABook(borrowedBookId):
52      cursor.execute(f"UPDATE Books SET Amount = Amount + 1 Where Id = {borrowedBookId}'")
```

cursor.execute(f"UPDATE Books SET Amount = Amount + 1 Where Id ='{borrowedBookId}'") – Zwrócenie książki do bazy danych poprzez dodanie 1 do Amount

9. Pobieranie czasu symulacji od użytkownika
    a) Funkcja pobierająca dane od użytkownika

```
54  def GetSimulationTime():
55      try:
56          monthsOfSimulation = int(input("Podaj liczbe miesiecy symulacji: "))
57          daysOfSimulation = int(input("Podaj liczbe dni symulacji: "))
58          hoursOfSimulation = int(input("Podaj liczbe godzin symulacji: "))
59          minutesOfSimulation = int(input("Podaj liczbe minut symulacji: "))
60          return float(minutesOfSimulation + hoursOfSimulation * 60 + daysOfSimulation * 480 + monthsOfSimulation * 14400)
61      except ValueError:
62          print("Podaj poprawna wartosc czasu.")
63          GetSimulationTime()
```

Program pobiera dane od użytkownika i przelicza czas symulacji na sekundy.
W przypadku podania błędnych danych program wykonuje funkcję *GetSimulationTime()* ponownie w celu uzyskania prawidłowych danych.

10. Generowanie wykresów
    a) Do generowania wykresów uzylismy biblioteki matplotlib oraz numpy.

```
cursor.execute(
    f"Select DayDATE, MonthDATE, count(*) as Amount FROM Clients Group By DayDate, MonthDate Order By DayDate,MonthDate ASC")
rows = cursor.fetchall()
objects = []
amountOfPeople = []
for row in rows:
    amountOfPeople.append(row[2])
    objects.append("M" + str(row[1]) + "D" + str(row[0]))
y_pos = np.arange(len(objects))
plt.title("Ilosc klientow w poszczegolne dni")
plt.xlabel("Miesiac oraz dzien")
plt.ylabel("Ilosc klientow")
plt.bar(y_pos, amountOfPeople, align='center', alpha=0.5)
plt.xticks(y_pos, objects)
plt.show()
```

Powyzej pokazany został fragment kodu odpowiedzialnego za rysowanie wykresu, w tym przypadku wykresu ukazującego ilość klientów w poszczególne dni. Siedem pierwszych linijek służy do uzyskania danych z bazy danych. Nastepnie za pomocą np.arrange tworzymy przestrzeń o danej długości z równymi odstępami. Linijki: plt.title, plt.xlabel, plt.ylabel, służą do dodania nazwy wykresu, osi x i y. Następnie plt.bar tworzy wykres z danych uzyskanych z bazy danych oraz przypisuje mu dane właściwości. Na końcu plt.show() ukazuje wykres.

11. Symulacja
    a) Generowanie klienta

```
######### SZANSA WZGLEDEM CZASU #########
if timeNow <= 20:
    chance = 25
if timeNow > 20 and timeNow <= 60:
    chance = 15
if timeNow > 60 and timeNow <= 120:
    chance = 20
if timeNow > 120 and timeNow <= 420:
    chance = 25
if timeNow > 420 and timeNow <= 480:
    chance = 40
if timeNow > minutesOfWorkInDay:
    chance = 0
########################################
#################### GENEROWANIE KLIENTA ####################
if random.randint(0, 99) < chance:
    newClient = Client(timeNow, day, month)
    if len(queue) > 0:
        newClient.queueTime = round(queue[len(queue) - 1].orderTime - (newClient.arrivalTime - queue[len(queue) - 1].arrivalTime), 2)
        newClient.orderTime = round(newClient.serviceTime + newClient.queueTime, 2)
    if (newClient.queueTime < 5):
        pleasedClients += 1
    else:
        unpleasedClients += 1
    queue.append(newClient)
    BorrowABook(newClient)
    ############################################################
```

Funkcja generuje obiekt klienta względem szansy w danej godzinie symulacji. Procentowa szansa wynika z rozkładu obciążenia biblioteki względem danych w *Google Maps*.

Do obiektu klienta przypisywane są wartości czasu wynikające z godziny symulacji o której przyszedł.

Pod koniec działania tej części programu do klienta przypisuje się ID wypożyczanej książki wykorzystując funkcję *BorrowABook()*.

b) Wgrywanie klienta do bazy danych

```
################################## WGRYWANIE KLIENTA DO BAZY DANYCH ##################################
name = "'" + newClient.name + "'"
lastname = "'" + newClient.lastname + "'"
time_ = timeNow
minute_ = timeNow % 60
hour_ = math.floor(time_ / 60)
if (newClient.queueTime < 5):
    pleased = "'" + 'YES' + "'"
else:
    pleased = "'" + 'NO' + "'"

cursor.execute("INSERT INTO Clients(Name,Lastname,MonthDATE,DayDATE,HourDATE,MinuteDATE,ArrivalTime,OrderTime,ServiceTime,QueueTime,Pleased,BorrowedBook,ReturnTime) "
               "VALUES (%s,%s,%d,%d,%d,%d,%d,%f,%f,%f,%s,%d,%d)"
               %(name,lastname,month,day,hour_,minute_,math.floor(newClient.arrivalTime),newClient.orderTime,newClient.serviceTime,newClient.queueTime,pleased,newClient.borrowedBook,newClient.returnTime))
cnxn.commit()
######################################################################################################
```

Obiekt klasy *Client* zostaje wgrany do bazy danych poprzez zapytanie języka MySQL.

c) Usuwanie klienta z kolejki

```
################### USUWANIE KLIENTA Z KOLEJKI ###################
if len(queue)>0:
    if(timeNow - queue[0].arrivalTime) > queue[0].orderTime:
        queue.pop(0)
##################################################################
```

Po minięciu czasu stania w kolejce przez klienta zostaje on usunięty z kolejki funkcją *pop().*

d) Oddawanie książki przez klienta

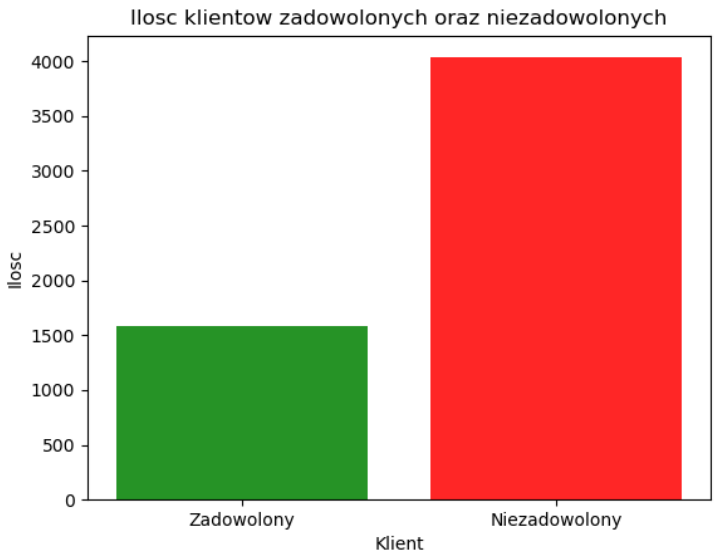
```
206         ##################### ZMIANA DNI I MIESIECY  ####################
207         if len(queue) == 0 and timeNow >= minutesOfWorkInDay:
208             day += 1
209             if day > 30:
210                 day = 1
211                 month += 1
212             cursor.execute(f"Select DayDATE, MonthDATE,ReturnTime,BorrowedBook From Clients WHERE BorrowedBook!=0")
213             rows = cursor.fetchall()
214             if rows:
215                 for row in rows:
216                     monthDelay = 0
217                     d = row[0]
218                     ret = row[2]
219                     if d + ret > 30:
220                         monthDelay = math.floor((d + ret) / 30)
221                         d = (d + ret) % 30
222                         ret = 0
223                     if [d + ret, row[1] + 1 + monthDelay] == [day, month]:
224                         ReturnABook(row[3])
225             if timeNow<minutesOfWorkInDay+0.5:
226                 print(str(round(timeNow))+"min" + " ================================================================================ " + str(round(runningTime))+"s")
227                 print("Nie ma nikogo w kolejce.")
228             startTime = time.time()
229             timer = 0
230         ############################################################
```

Część programu która sprawdza opóźnienie klienta w oddaniu książki. Jeśli czas zgadza się z czasem symulacji książka zostaje oddana funkcją *ReturnABook().*

12. Statystyka

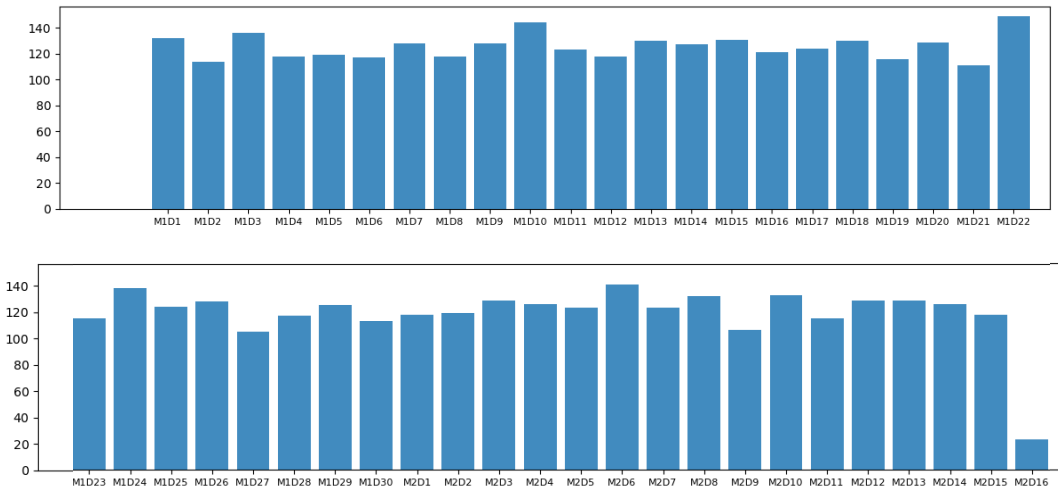
Po przeprowadzeniu symulacji przez około 1 miesiąc i 16 dni otrzymaliśmy następujące wyniki.

a) Wykres słupkowy zadowolonych i niezadowolonych klientów



Można zauważyć dysproporcję w ilości klientów zadowolonych i niezadowolonych. Ilość klientów zadowolonych jest zauważalnie mniejsza. Wynika to z ilości osób które przyszły do biblioteki w małym odstępie czasu. Zależność tą można zmniejszyć poprzez zmniejszenie maksymalnej możliwej ilości klientów w danej godzinie działania biblioteki

b) Ilość klientów w poszczególne dni



Na powyższych wykresach można zauważyć że dziennie bibliotekę średnio odwiedza około 130 osób. Jedynie ostatniego dnia symulacji ilość osób jest dużo mniejsza ponieważ ostatniego dnia symulacja trwała jedynie 4 godziny.

c) Stosunek udanych wypożyczeni do nieudanych

Wykres ilustruje ilość klientów którym udało się wypożyczyć książkę oraz klientów którym nie udało się wypożyczyć książki z powodu braku jej w zbiorze. Ilość klientów z niewypożyczoną książką można zmniejszyć poprzez skrócenie czasu oddania książki lub zwiększenie ilości książek w zbiorze.

Wykres ilustruje ilość klientów którym udało się wypożyczyć książkę (o *ID* generowanym pseudolosowo) oraz klientów którym nie udało się wypożyczyć książki z powodu braku jej w zbiorze (*Amount=0)*.