

Proiect final Cloud Computing

1. Descrierea temei

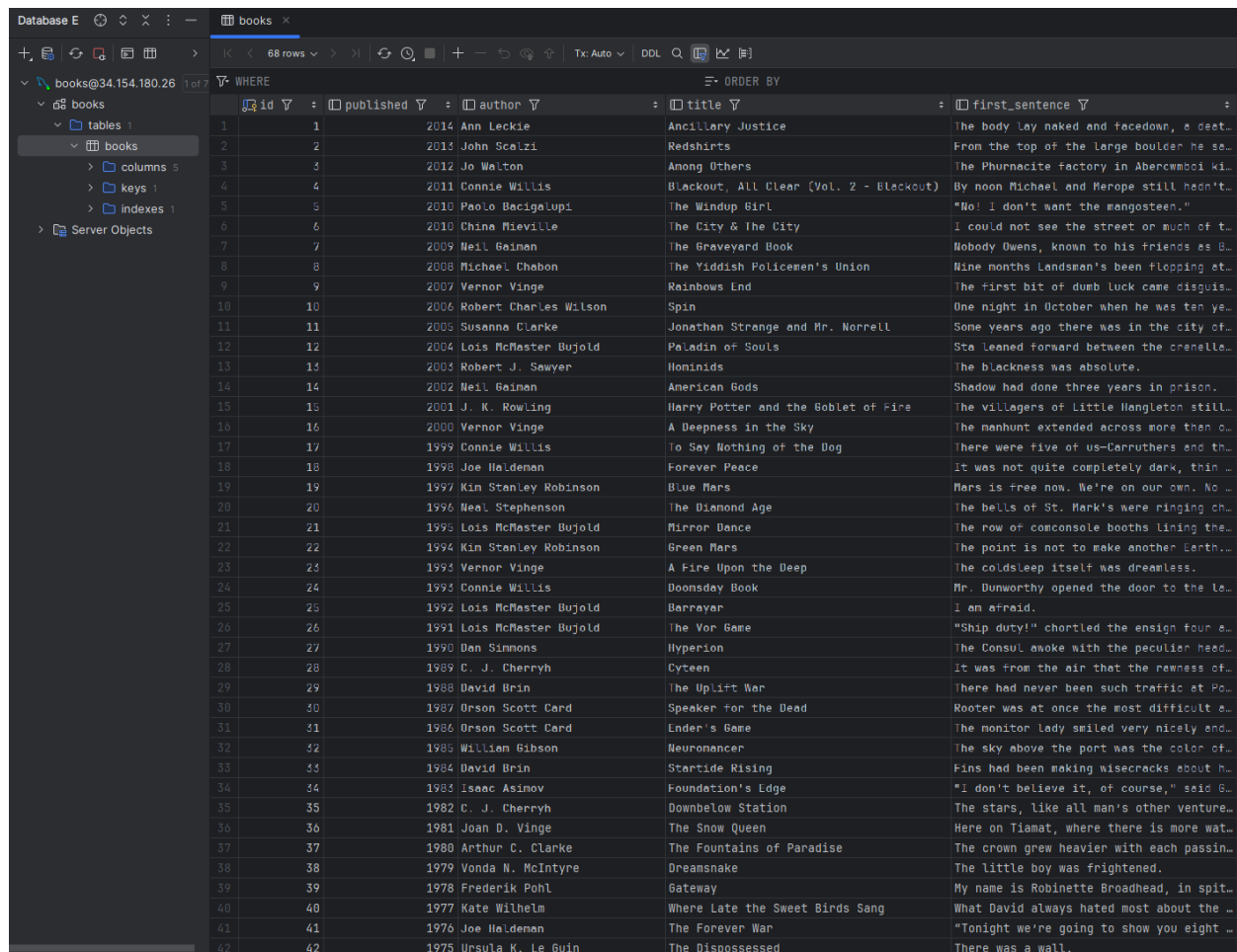
Proiectul constă în dezvoltarea unei interfețe de programare a aplicațiilor (API) simple folosind Python și Flask pentru a gestiona date despre cărți. Scopul este de a permite utilizatorilor să acceseze și să interogheze o bază de date despre cărți pentru a obține informații relevante.

Acest proiect pornește de la API-ul open source disponibil pe GitHub, [simple-flask-api](#). Pornind de la implementarea disponibilă cu o bază locală de date SQLite și metode(GET/POST) pentru interacțiuni cu baza de date, noi dorim să aducem această aplicație la un stadiu care să fie ‘user-friendly’.

2. Descrierea modelului de date utilizat

Modelul de date folosit este simplu și eficient. Baza noastră de date books conține un tabel books care are următoarea structură ușor de înțeles pentru un utilizator al unei aplicații din universul cărților:

- id: Un identificator care ar trebui să fie unic pentru fiecare carte.
- title: Titlul cărții.
- author: Numele autorului cărții.
- published: Anul publicării cărții.
- first_sentence: Prima propoziție a cărții.



id	published	author	title	first_sentence
1	2014	Ann Leckie	Ancillary Justice	The body lay naked and facedown, a dead...
2	2014	John Scalzi	Redshirts	From the top of the large boulder he sa...
3	2012	Jo Walton	Among Others	The Phurnacite factory in Abercwmboi ki...
4	2011	Connie Willis	Blackout, All Clear (Vol. 2 - Blackout)	By noon Michael and Herope still hadn't...
5	2010	Paolo Bacigalupi	The Windup Girl	"No! I don't want the mangosteen."
6	2010	China Mieville	The City & The City	I could not see the street or much of t...
7	2009	Neil Gaiman	The Graveyard Book	Nobody Owens, known to his friends as B...
8	2008	Michael Chabon	The Yiddish Policemen's Union	Nine months Landsman's been flopping at...
9	2007	Vernor Vinge	Rainbows End	The first bit of dumb luck came disguis...
10	2006	Robert Charles Wilson	Spin	One night in October when he was ten ye...
11	2005	Susanne Clarke	Jonathan Strange and Mr. Norrell	Some years ago there was in the city of...
12	2004	Lois McMaster Bujold	Paladin of Souls	Ste leaned forward between the crenella...
13	2003	Robert J. Sawyer	Hominiids	The blackness was absolute.
14	2002	Neil Gaiman	American Gods	Shadow had done three years in prison.
15	2001	J. K. Rowling	Harry Potter and the Goblet of Fire	The villagers of Little Hangleton still...
16	2000	Vernor Vinge	A Deepness in the Sky	The manhunt extended across more than o...
17	1999	Connie Willis	To Say Nothing of the Dog	There were five of us-Carruthers and th...
18	1998	Joe Maldenan	Forever Peace	It was not quite completely dark, thin ...
19	1997	Kin Stanley Robinson	Blue Mars	Mars is free now. We're on our own. No ...
20	1996	Neal Stephenson	The Diamond Age	The bells of St. Mark's were ringing ch...
21	1995	Lois McMaster Bujold	Mirror Dance	The row of commonplace booths lining the...
22	1994	Kin Stanley Robinson	Green Mars	The point is not to make another Earth...
23	1993	Vernor Vinge	A Fire Upon the Deep	The coldsleep itself was dreamless.
24	1993	Connie Willis	Doomsday Book	Mr. Dunworthy opened the door to the la...
25	1992	Lois McMaster Bujold	Barrovay	I am afraid.
26	1991	Lois McMaster Bujold	The Vor Game	"Ship duty!" chortled the ensign four a...
27	1990	Dan Simmons	Hyperion	The Consul awoke with the peculiar head...
28	1989	C. J. Cherryh	Cyteen	It was from the air that the rewness of...
29	1988	David Brin	The Uplift War	There had never been such traffic at Po...
30	1987	Orson Scott Card	Speaker for the Dead	Rooter was at once the most difficult a...
31	1986	Orson Scott Card	Ender's Game	The monitor lady smiled very nicely and...
32	1985	William Gibson	Neuromancer	The sky above the port was the color of...
33	1984	David Brin	Startide Rising	Fins had been making wisecracks about h...
34	1983	Isaac Asimov	Foundation's Edge	"I don't believe it, of course," said G...
35	1982	C. J. Cherryh	Downbelow Station	The stars, like all man's other venture...
36	1981	Joan D. Vinge	The Snow Queen	Here on Tiamat, where there is more wat...
37	1980	Arthur C. Clarke	The Fountains of Paradise	The crown grew heavier with each passin...
38	1979	Vonda N. McIntyre	Dreamsnake	The little boy was frightened.
39	1978	Frederik Pohl	Gateway	My name is Robinette Broadhead, in spit...
40	1977	Kate Wilhelm	Where Late the Sweet Birds Sang	What David always hated most about the ...
41	1976	Joe Maldenan	The Forever War	"Tonight we're going to show you eight ...
42	1975	Ursula K. Le Guin	The Dispossessed	There was a wall.

3. Stocarea datelor

Soluția aleasă pentru stocarea datelor este o instanță MySQL 8.0 configurată pe o mașină virtuală în Google Cloud. MySQL este un sistem de gestionare a bazelor de date relaționale (RDBMS). Am ales MySQL datorită următoarele avantaje: performanței, scalabilității și robusteții sale și flexibilitate pentru gestionarea datelor și interogarea acestora în mod eficient.

În plus, folosind Google Cloud pentru mașina virtuală care stochează instanța MySQL ne sunt asigurate accesibilitate și securitate. Platforma Google Cloud oferă servicii fiabile și scalabile de gestionare a bazelor de date, facilitând administrarea și monitorizarea acestora. Singurul dezavantaj apare în momentul în care mașina virtuală e oprită și apoi repornită și astfel este nevoie să fie modificat hostname-ul în codul aplicației noastre.

4. Testarea unei operații de extragere a datelor

Pentru a testa extragerea datelor din baza de date MySQL, am folosit Python și Flask. Am implementat o funcționalitate similară cu cea din proiectul inițial, dar adaptată pentru a lucra cu MySQL. Iată un exemplu de cod pentru extragerea tuturor cărților:

```
from flask import Flask, request, jsonify, render_template
import pymysql
import getpass
import os

# Init app
app = Flask(__name__)

# Transform results into dictionaries
1 usage
def dict_factory(cursor, row):
    d = {}
    for idx, col in enumerate(cursor.description):
        d[col[0]] = row[idx]
    return d

# MySQL connection info from environment variables
db_host = os.getenv('DB_HOST', '34.154.180.26')
db_user = os.getenv('DB_USER', 'sergiujoandrea')
db_password = os.getenv('DB_PASSWORD', '240100')

# Connect to MySQL
3 usages
def connect_to_mysql():
    return pymysql.connect(host=db_host, user=db_user, password=db_password, port=3306, database='books'
                           , charset='utf8', cursorclass=pymysql.cursors.DictCursor)
```

Avram Alin-Petru
Joandrea Sergiu-Cătălin

```
# Home page of the app
@app.route(rule: '/', methods=['GET'])
def home():
    connection = None
    try:
        connection = connect_to_mysql()
        with connection.cursor() as cursor:
            cursor.execute("SELECT user from mysql.user;")
            result = cursor.fetchall()
            user = result[0]['user']
            return render_template(template_name_or_list: 'home.html', connected=True, user=user)
    except Exception as e:
        return render_template(template_name_or_list: 'home.html', connected=False, error=str(e))
    finally:
        if connection:
            connection.close()
```

```
# Endpoint to filter books based on certain criteria
@app.route(rule: '/api/v2/resources/books', methods=['GET'])
def api_filter():
    query_parameters = request.args

    title = query_parameters.get('title')
    published = query_parameters.get('published')
    author = query_parameters.get('author')
    first_sentence = query_parameters.get('first_sentence')

    query = 'SELECT * FROM books WHERE'
    to_filter = []

    if title:
        query += ' title LIKE %s AND'
        to_filter.append('%' + title + '%')

    if published:
        query += ' published LIKE %s AND'
        to_filter.append('%' + published + '%')

    if author:
        query += ' author LIKE %s AND'
        to_filter.append('%' + author + '%')

    if first_sentence:
        query += ' first_sentence LIKE %s AND'
        to_filter.append('%' + first_sentence + '%')

    if not (title or published or author or first_sentence):
        return page_not_found(404)

    query = query[:-4] + ';'

    connection = None
    try:
        connection = connect_to_mysql()
        with connection.cursor() as cursor:
            cursor.execute(query, to_filter)
            results = cursor.fetchall()
            return jsonify(results)
    except Exception as e:
        return f"Error: {str(e)}"
    finally:
        if connection:
            connection.close()
```

Avram Alin-Petru
Joandrea Sergiu-Cătălin

Rezultatul obținut:

Home Page

Connected to MySQL. User: sergiujoandrea

Author:

Published:

Title:

First Sentence:

Home Page

Connected to MySQL. User: sergiujoandrea

Author:

Published:

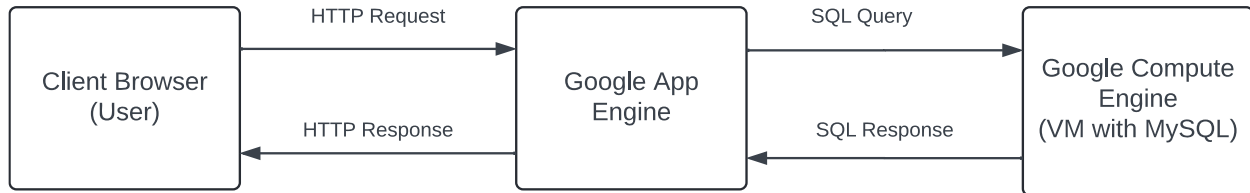
Title:

First Sentence:

Pretty-print ☒

```
[
  {
    "author": "Paolo Bacigalupi",
    "first_sentence": "\"No! I don't want the mangosteen.\"",
    "id": 5,
    "published": 2010,
    "title": "The Windup Girl"
  },
  {
    "author": "China Mieville",
    "first_sentence": "I could not see the street or much of the estate.",
    "id": 6,
    "published": 2010,
    "title": "The City & The City"
  }
]
```

5. Diagrama bloc



Componentele și fluxurile de date

Client Browser: Acest bloc reprezintă utilizatorul care accesează aplicația web prin intermediul unui browser.

Google App Engine: Acesta este serviciul gestionat de GCP care găzduiește și rulează aplicația Flask. Este responsabil pentru gestionarea cererilor HTTP și pentru apelarea interogărilor către baza de date MySQL.

Google Compute Engine (VM cu MySQL): Acesta este un VM pe care se află instanța MySQL. Aplicația Flask din Google App Engine se conectează la acest VM pentru a efectua interogări în baza de date.

Descrierea fluxului

Fluxul 1: Utilizatorul face o cerere HTTP către aplicația găzduită pe Google App Engine.

Fluxul 2: Google App Engine primește cererea și face o interogare SQL către baza de date MySQL pe Google Compute Engine.

Fluxul 3: Google Compute Engine procesează interogarea și trimite răspunsul înapoi la Google App Engine.

Fluxul 4: Google App Engine procesează răspunsul și trimite răspunsul HTTP înapoi la utilizator.

6. Maturitatea aplicației din perspectiva arhitecturii native cloud

Aplicația noastră utilizează serviciile GCP astfel încât să beneficieze de scalabilitate, gestionare ușoară și fiabilitate. Analizând propria aplicație pe cele trei axe ale arhitecturii native cloud obținem următoarele:

Avram Alin-Petru

Joandrea Sergiu-Cătălin

- **Scalabilitate:** Google App Engine permite scalarea automată a aplicației în funcție de cerere, ceea ce înseamnă că aplicația noastră poate face față unui număr mare de utilizatori fără intervenție manuală.

- **Gestionare ușoară:** Utilizarea serviciilor gestionate de GCP, cum ar fi App Engine, reduce complexitatea administrării infrastructurii. De asemenea, GCP oferă monitorizare și alerte integrate pentru gestionarea performanței aplicației.

- **Fiabilitate:** Baza de date MySQL pe Google Compute Engine poate fi configurată pentru backup-uri automate și replicare, asigurând astfel durabilitatea și disponibilitatea datelor.

Din punct de vedere al SLA putem afirma următoarele:

- Google App Engine: Google garantează un SLA de 99.95% pentru aplicațiile găzduite pe App Engine, ceea ce înseamnă că timpul maxim de nefuncționare este de aproximativ 21.56 minute pe lună. (sursa: <https://cloud.google.com/appengine/sla>)

- Google Compute Engine: Google oferă un SLA de 99.99% pentru instanțele Compute Engine, asigurând astfel o disponibilitate foarte mare pentru baza de date MySQL. (sursa: <https://cloud.google.com/compute/sla>)