OCEANCROSS

Planificarea croazierelor

Arhitectura sistemului

Nume proiect	OceanCross
Nume document	Arhitectura sistemului / Architectural Design Document
Autori	Bărbuță Delia Elena, Grabovschi Adrian, Pristanda Amalia Maria, Ungureanu Mihai Vlăduț, Vecliuc Draia Teodora

1 Introducere

1.1 Scopul documentului

Acest document oferă o imagine de ansamblu asupra arhitecturii sistemului. Scopul lui este atât de a descrie diferitele componente ale proiectului software, cât și de a transmite deciziile arhitecturale semnificative care au fost luate.

1.2 Scurtă descriere a proiectului

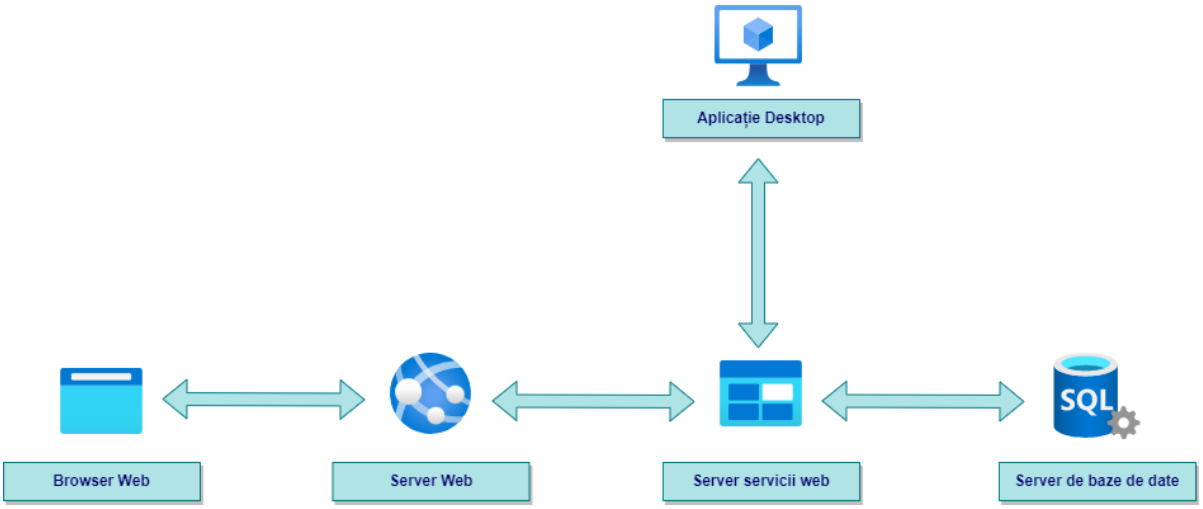
OceanCross vine în ajutorul persoanelor care doresc să-și aleagă croazieră perfectă și organizatorilor de astfel de vacanțe. În cadrul acestui proiect, utilul se îmbină cu plăcutul, persoanele fizice putând să aleagă atât croaziera care li se potrivește, cât și activitățile din timpul staționării în diverse porturi, iar organizatorii își pot dispune ofertele într-un mediu sigur: totul într-o interfață web ușor de utilizat.

1.3 Definiții și abrevieri

- Interfața web = interfața de interacțiune cu utilizatorul ce este accesată prin intermediul Web-ului

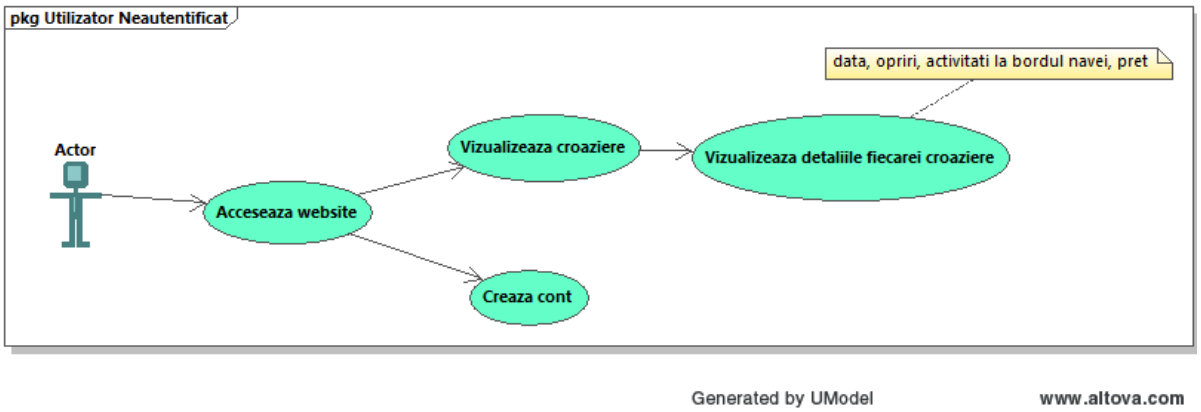
2 Prezentarea sistemului

Diagrama din figură prezintă interacțiunea dintre componentele principale ale sistemului:

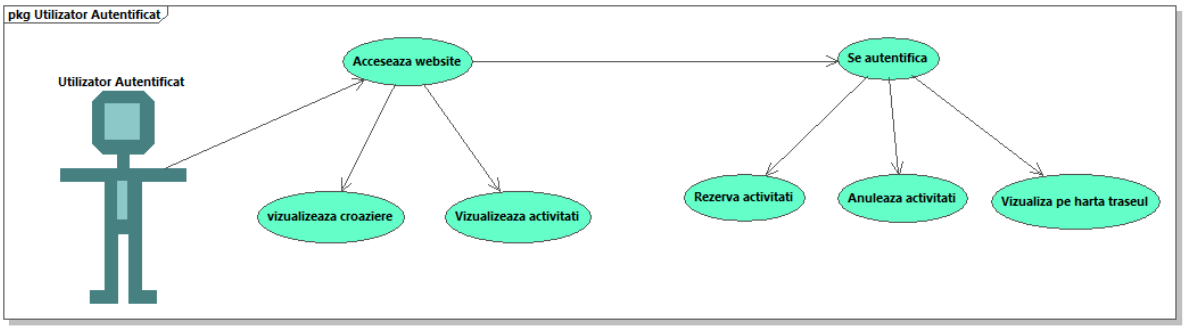


În cele ce urmează sunt redate diagramele UML Use Case, pentru fiecare tip de utilizator din cadrul aplicației:

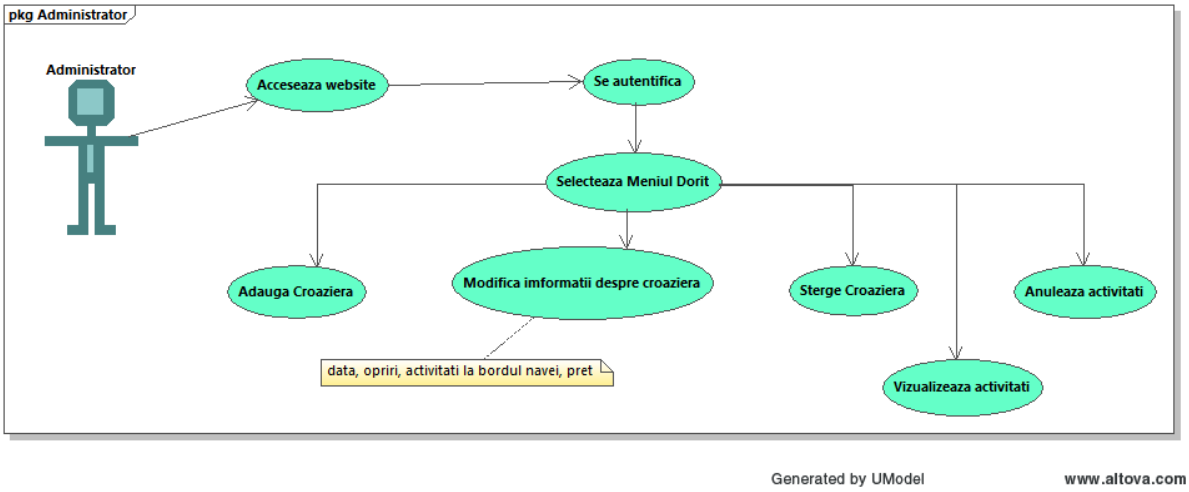
- Utilizator neautentificat



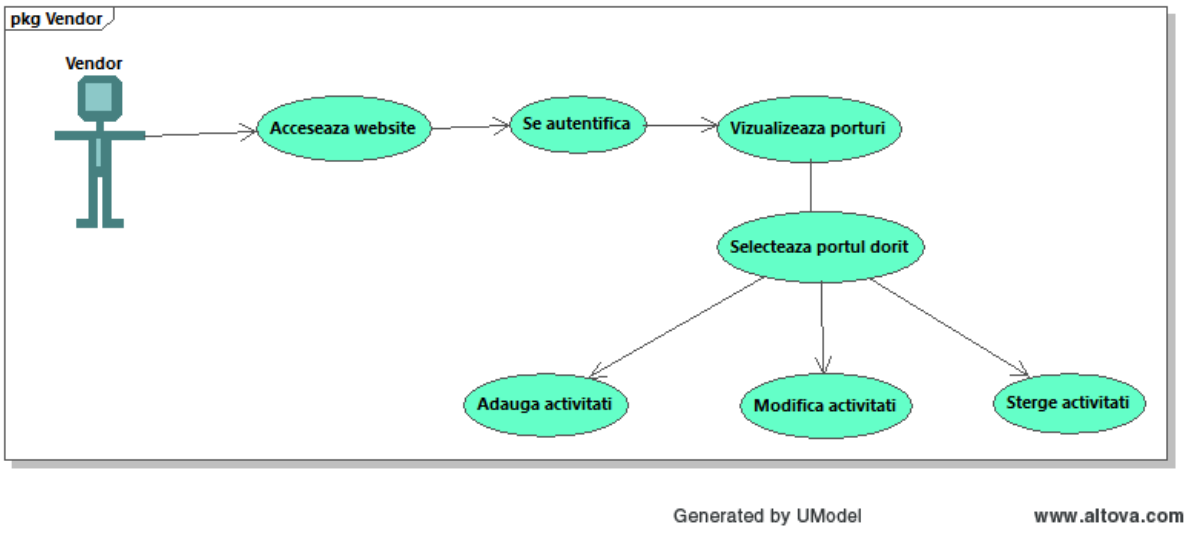
- Utilizator autentificat



- Administrator



- Vendor



3 Arhitectura top-level

3.1 Identificarea componentelor

Browserul Web este o aplicație software folosită pentru accesarea site-urilor web. Când un utilizator solicită o pagină web de pe un anumit website, browserul preia fișierele de la un server web și apoi afișează pagina pentru utilizator. Scopul unui browser web este de a prelua conținut de pe World Wide Web sau din local Storage și de-al afișa pe dispozitivul utilizatorului.

Serverul Web este alcătuit din componente care controlează modul în care utilizatorii accesează fișierele găzduite. Sarcina principală a unui server web este de a afișa conținutul site-ului web prin stocarea, procesarea și livrarea paginilor web către utilizatori. Aceasta componenta utilizează în cadrul proiectului tehnologia Angular și limbajul TypeScript.

Serverul servicii web este o aplicație software ce expune o serie de endpointuri, utilizând un sistem de mesaje de tip cerere-răspuns, de obicei exprimate în JSON sau XML. Un endpoint este un capăt al unui canal de comunicare, cel mai des fiind expus prin intermediul unui protocol bazat pe HTTP. Sintagma servicii Web referă o colecție de standarde ce adresează interoperabilitatea. Aceste standarde definesc suita de protocoale și interfețele ce vor fi utilizate de oricare două entități ce vor participa într-un schimb de mesaje. Pentru implementarea acestor servicii este utilizat framework-ul Spring Boot, respectiv limbajul Java.

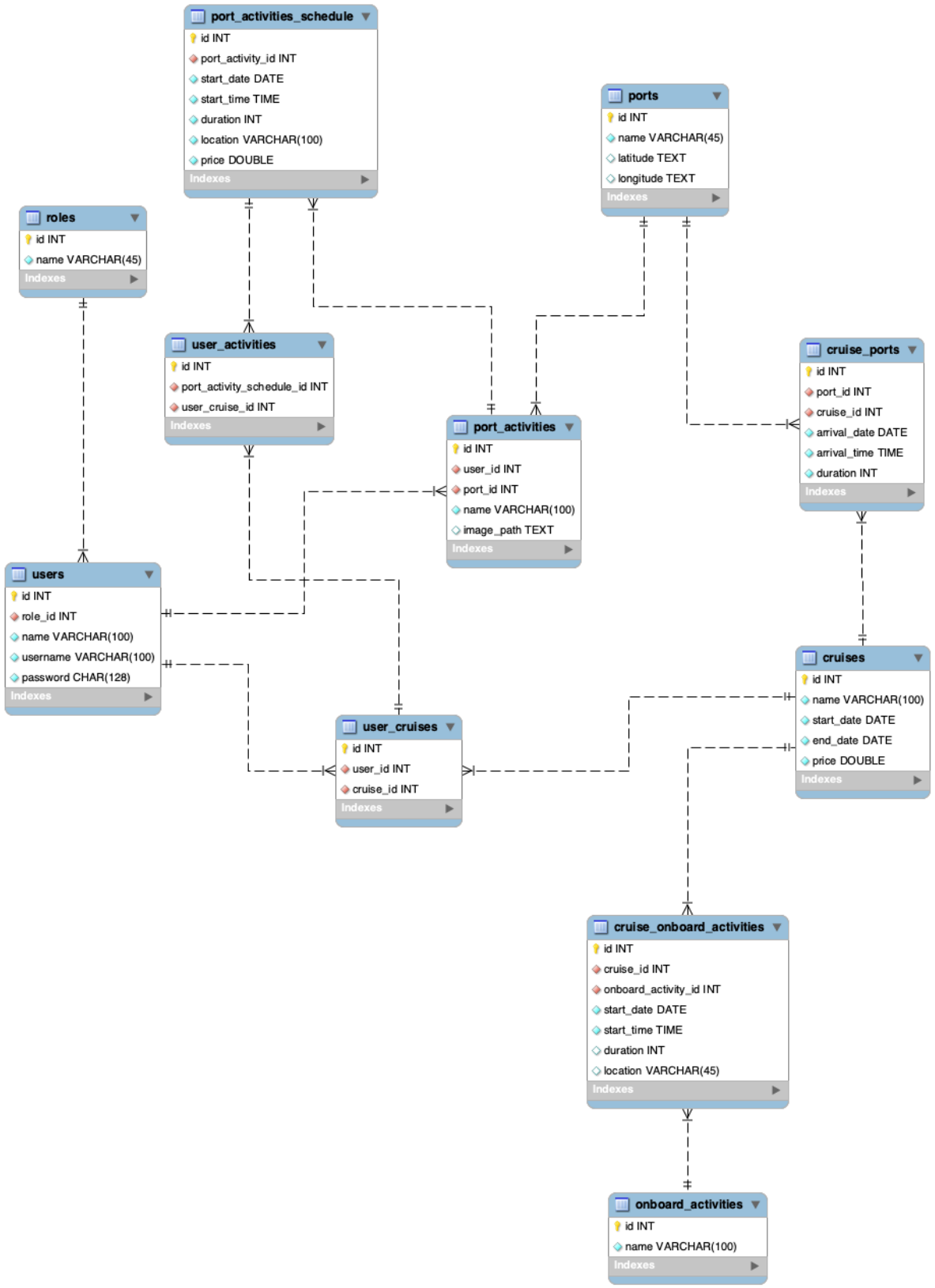
Serverul de baze de date este un server ce furnizează servicii de baze de date altor programe, așa cum este definit în modelul client-server. În cadrul acestui server sunt executate interogari si preluate rezultate. Un server de baze de date este necesar, deoarece acesta este responsabil de persistența datelor, ceea ce reprezintă un aspect cheie al oricărei aplicații. În cadrul sistemului, serverul va fi de tip MySQL, pentru a lucra cu baze de date relaționale și pentru că acesta oferă un suport cuprinzător pentru dezvoltarea aplicațiilor web.

Aplicație Desktop este un program software care rulează local pe dispozitive computerizate. Ele nu sunt accesibile dintr-un browser, cum ar fi aplicațiile web și necesită implementare pe un computer personal sau laptop.

3.2 Relațiile și comunicarea dintre componente

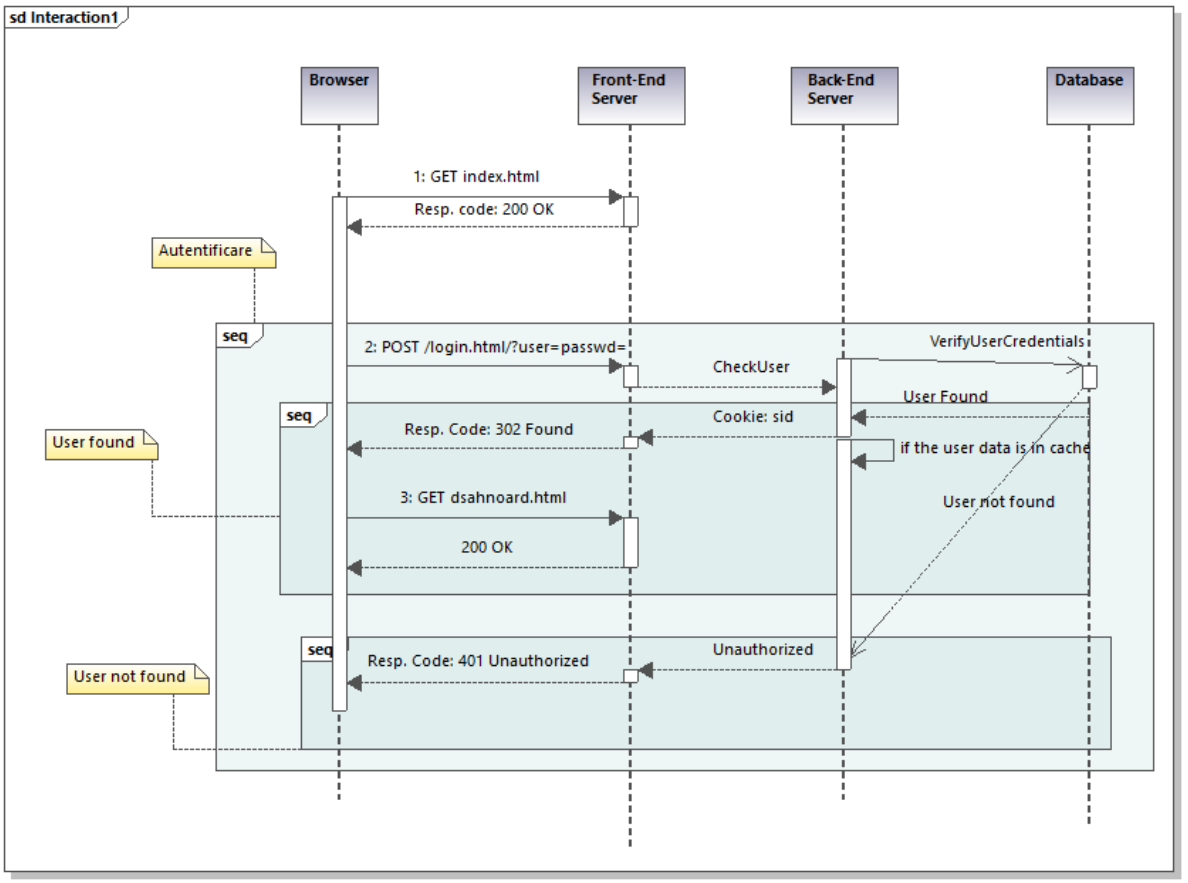
3.2.1. Diagrama ER

Diagrama ER (Entity Relationship Diagram) este utilă pentru a vizualiza relațiile dintre entitățile unui sistem și a explica structura logica a bazei de date. Tabelele redate în aceasta diagrama vor fi implementate folosind bază de date de tip MySQL.



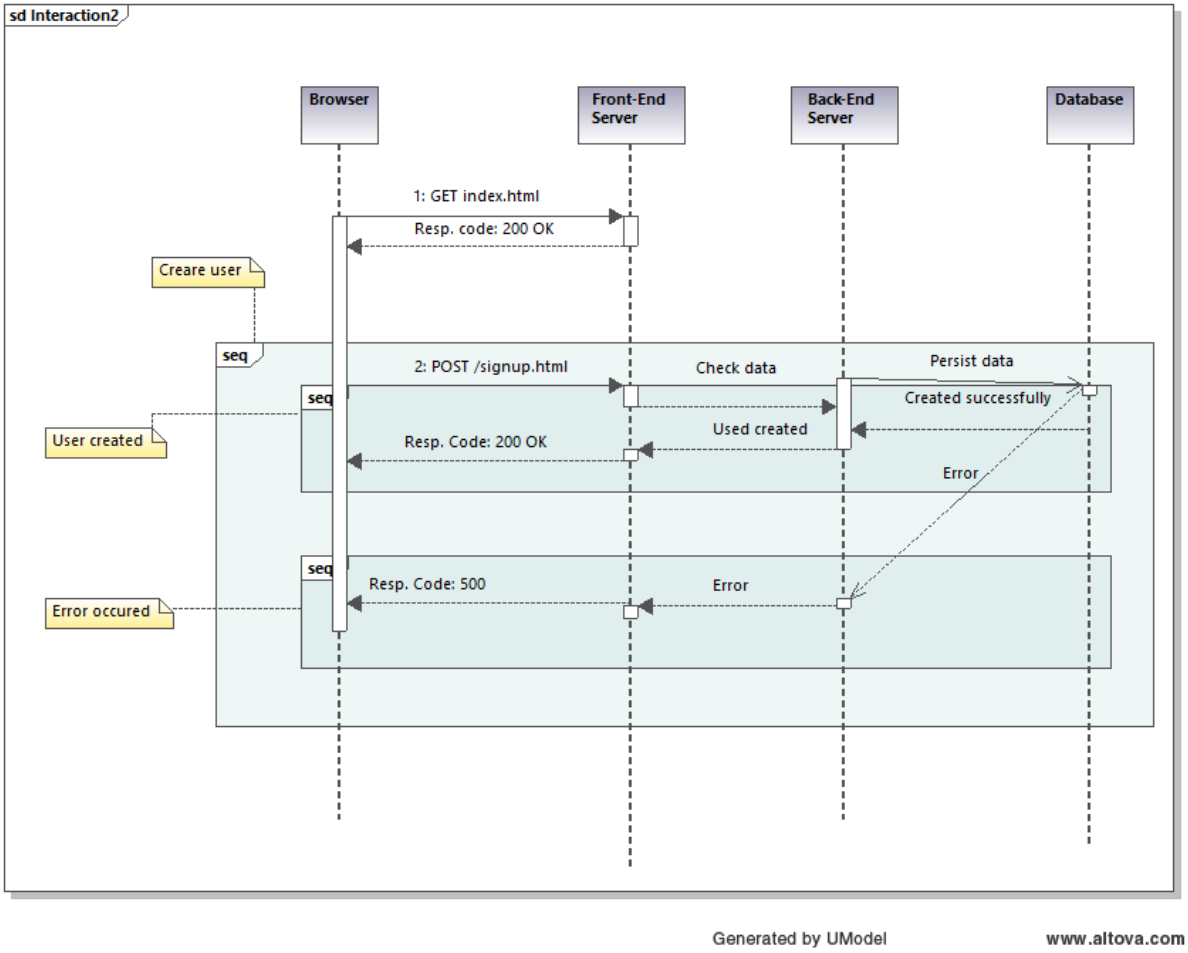
Următoarele diagrame de secvența descriu cateva flow-uri identificate în cadrul sistemului.

Autentificare:

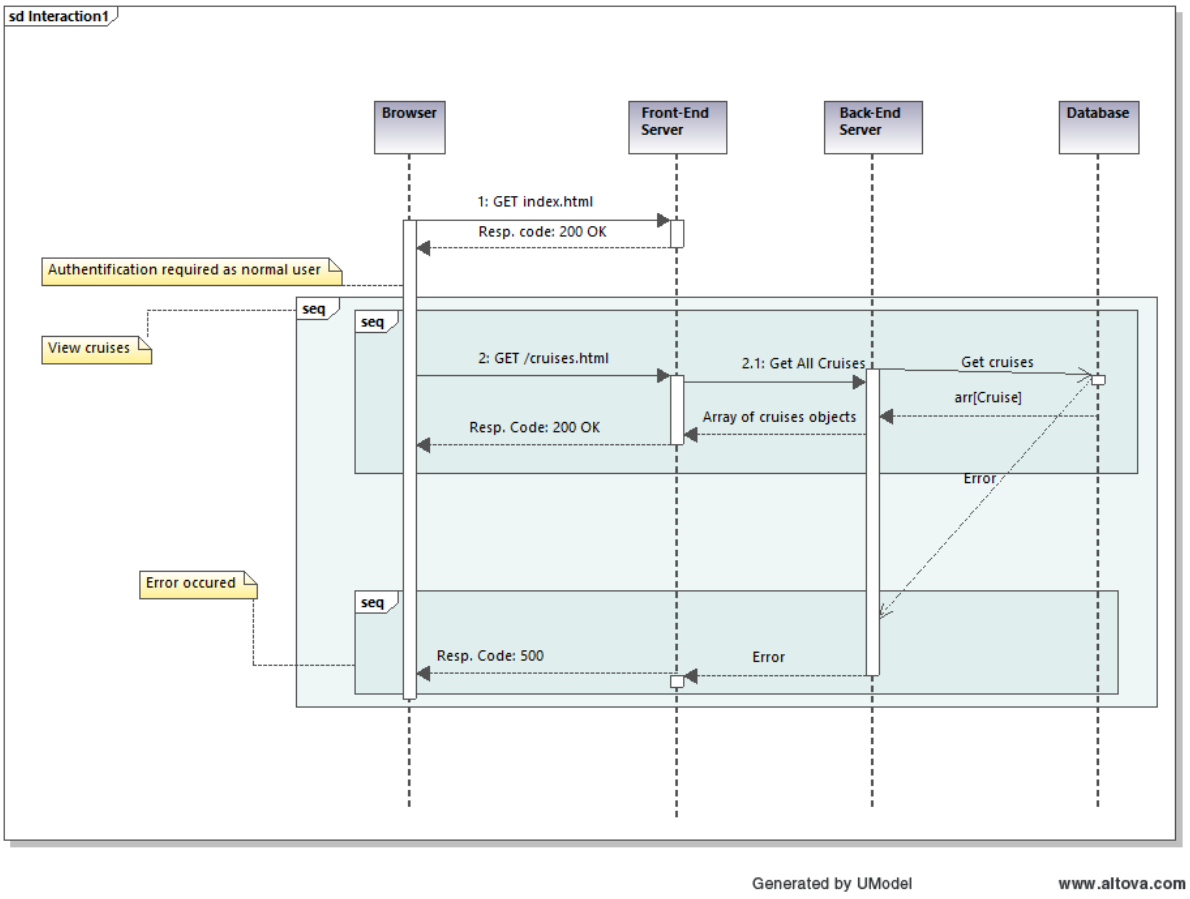


Generated by UModel www.altova.com

Creare user: (presupunem că ruta /signup este mapată corespunzător)



Vizualizare croaziere: (presupunem că ruta /cruises este mapata corespunzător)



4 Descrierea componentelor

Browserul web

Un browser web este “un program folosit pentru a accesa site-uri sau informații într-o rețea (cum ar fi WORLD WIDE WEB) ”. In general, este necesara interactiunea cu utilizatorul pentru a specifica browserului ce pagina web dorim sa vizualizam. Browserul web folosește un protocol de comunicare , de obicei HTTP sau HTTPS, prin intermediul căruia preia o resursa și o expune într-un format ușor de vizualizat. Browserul web va fi responsabil de interacțiunea cu utilizatorul și de informațiile expuse acestuia.

Serverul web

În cadrul acestui proiect se va folosi un server web local, webpack-dev-server, și ca platforma de dezvoltare va fi utilizat Angular. Acesta este construit în TypeScript și oferă un cadru bazat pe componente pentru construirea de aplicații web scalabile, o colecție de biblioteci bine integrate care acoperă o mare varietate de caracteristici, inclusiv rutare, gestionare a formularelor, comunicare client-server. Ca modalitate de încărcare a paginilor acesta folosește SPA, Single Page Application. SPA sunt aplicații web care încarcă o singură pagină HTML și doar o parte a paginii, în loc de întreaga pagină, este actualizată cu fiecare click al

mouse-ului. Pagina nu se reîncarcă și nu transferă controlul către o altă pagină în timpul procesului. Acest lucru asigură performanță ridicată și încărcarea rapidă a paginilor.

Serverul de servicii web

În cadrul sistemului, se folosește o arhitectura **SOA (Service Oriented Architecture)**, ce reprezinta un model de dezvoltare de aplicații care presupune că functionalitatea dorită poate fi obtinut prin conectarea unor module software slab legate (denumite servicii). Altfel spus, o aplicație dezvoltată utilizând SOA nu reprezintă altceva decât o colecție de servicii care implementează functionalitatea dorită.

Modulele serverului de servicii web sunt detaliate în cele ce urmează.

Modulul UserManangement: este componenta ce se ocupă de crearea de conturi noi, deservește functionalitatile de login, logout și stocarea datelor despre utilizatori în baza de date. Acesta primește ca input date de la utilizator, comunică cu tabela users, prin **repository** și returnează fie un mesaj de succes, fie un token, fie un mesaj de eroare în cazul în care nu s-a reușit crearea unui nou cont sau validarea credențialelor nu a reușit.

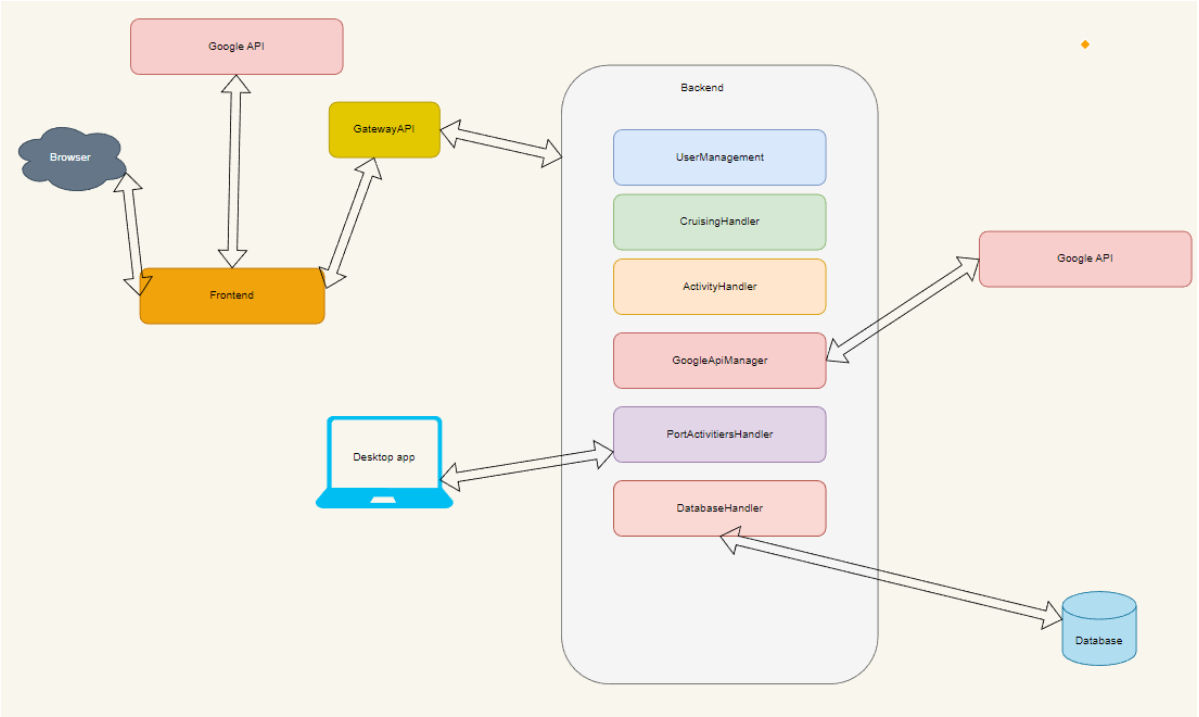
Modulul CruisingHandler: este componenta ce se ocupă de gestionarea croazierelor (creare, extragere de informatii, actualizare, ștergere). Ea primește ca date de input informații referitoare la croazieră, comunică cu tabelele ports, cruise_ports, cruises ,onboard_activities și cruise_onboard_activities, prin **repository** și returnează un mesaj ce indică starea cererii (dacă aceasta a fost procesata cu succes sau nu).

Modulul PortActivitiesHandler: este componenta ce se ocupă de expunerea și gestionarea activităților puse la dispoziție de vendori: adăugarea de activități în porturi, actualizarea sau ștergerea acestora. Această componentă primește ca input portul, activitatea și informații despre aceasta, comunică cu baza de date - tabelele ports, ports_activities și ports_activities_schedule - prin intermediul **repository** și returnează un mesaj ce indică starea procesării.

Modulul ActivityHandler: este componenta ce se ocupă de gestionarea informațiilor unui utilizator cu privire la croazierele la care acesta vrea să participe și lista de activități pe care le va desfășura in porturi, componenta va comunica cu tabelele user_cruises și user_activities, prin intermediul **repository** și va returna un mesaj ce conține informații referitoare la statusul de procesare a acțiunii dorite.

Modulul GoogleApiHandler: este o interfață pentru gestionarea serviciului GoogleAPI.

Modulul DatabaseHandler: este o componentă ce încapsulează functionalitatile de comunicare cu baza de date. Această componentă este oferă o abstractizare pentru funcțiile CRUD facilitând interacțiunea backend - database.



Protocolul HTTP:

Protocolul HTTP(Hyper Text Markup Language) este un protocol de tip text folosit pentru a transmite informații între un program de navigare (browser web) și un server .

Metoda GET : solicită o reprezentare a resursei specificate. Solicitările care utilizează GET ar trebui folosite numai pentru a solicita date.

Metoda POST : este folosită pentru a trimite date ce vor fi procesate la server, pentru a crea sau actualiza o resursă.

Metoda PUT : este folosită pentru a trimite date către server, pentru a crea sau actualiza o resursa.

Diferența dintre POST și PUT este ca toate cererile PUT sunt idempotente, adică apelarea aceleiași cereri PUT de mai multe ori va produce mereu același rezultat, însă apelarea cererii POST în mod repetat are efectul secundar de a creea aceeași resursa de mai multe ori.

Metoda DELETE : este folosită pentru ștergerea unei resurse specificate.

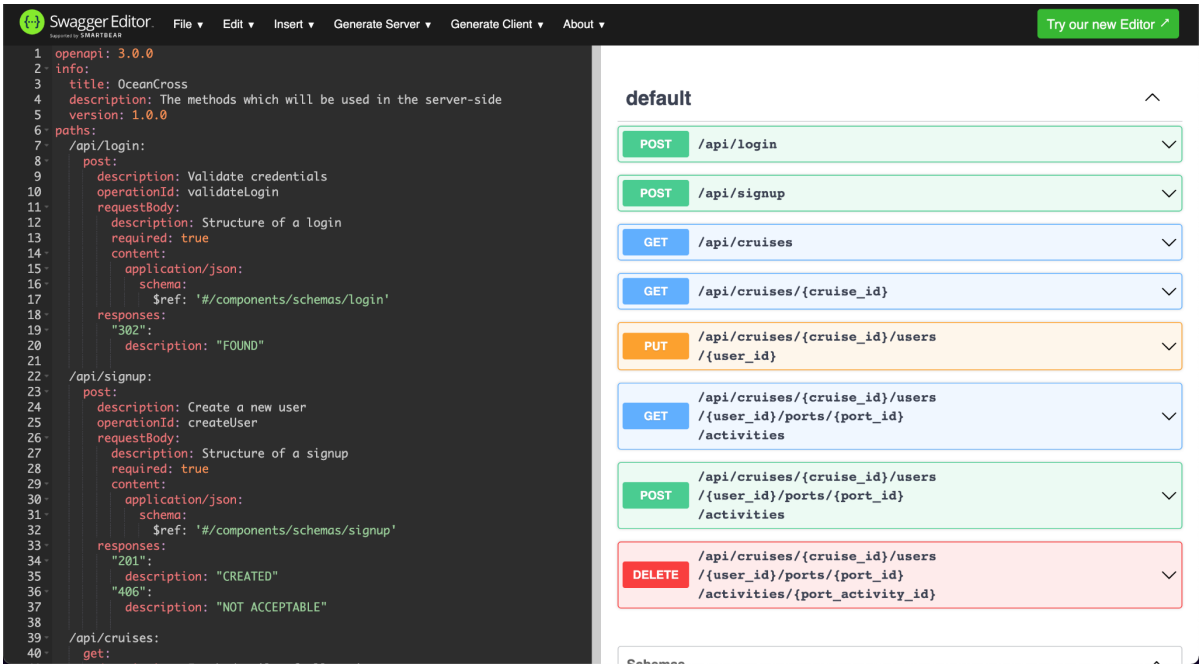
Swagger este un editor open-source folosit pentru definirea și documentarea API-urilor bazate pe protocolul HTTP, ce folosesc specificațiile OpenAPI. În contextul acestui sistem, Swagger Editor a fost folosit pentru a descrie metodele ce vor fi puse la dispoziția unui utilizator autentificat.

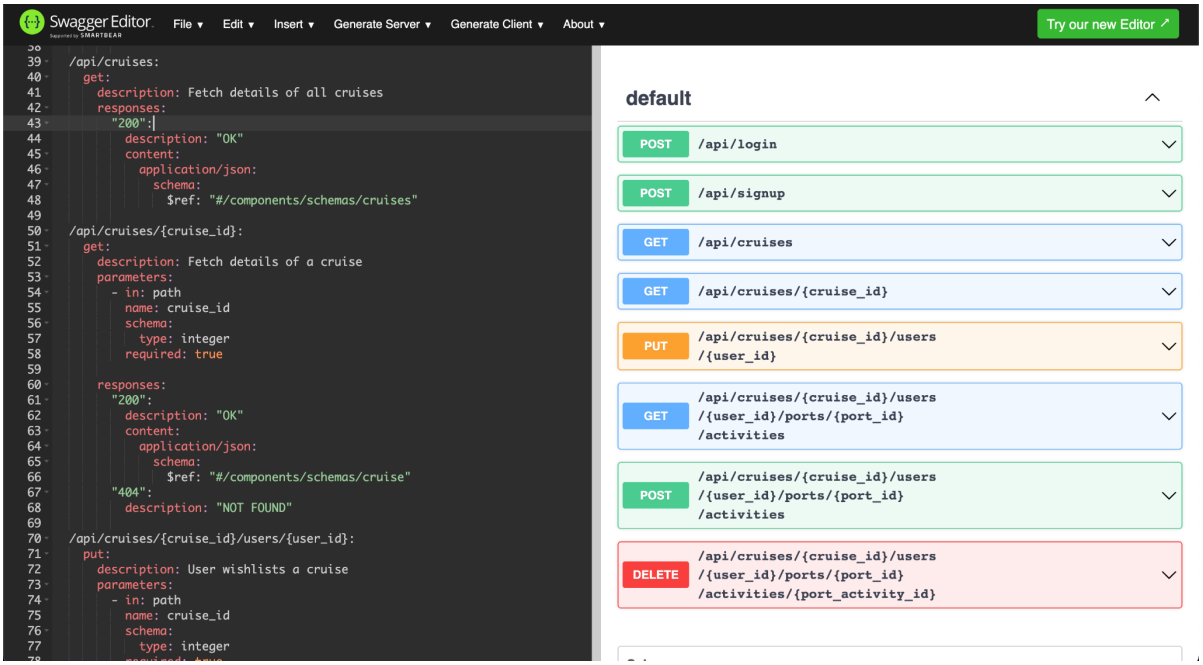
Pentru vizualizarea informațiilor, precum lista completă a croazierelor, detaliile unei singure croaziere sau lista activităților disponibile într-un port, este utilizată metoda GET.

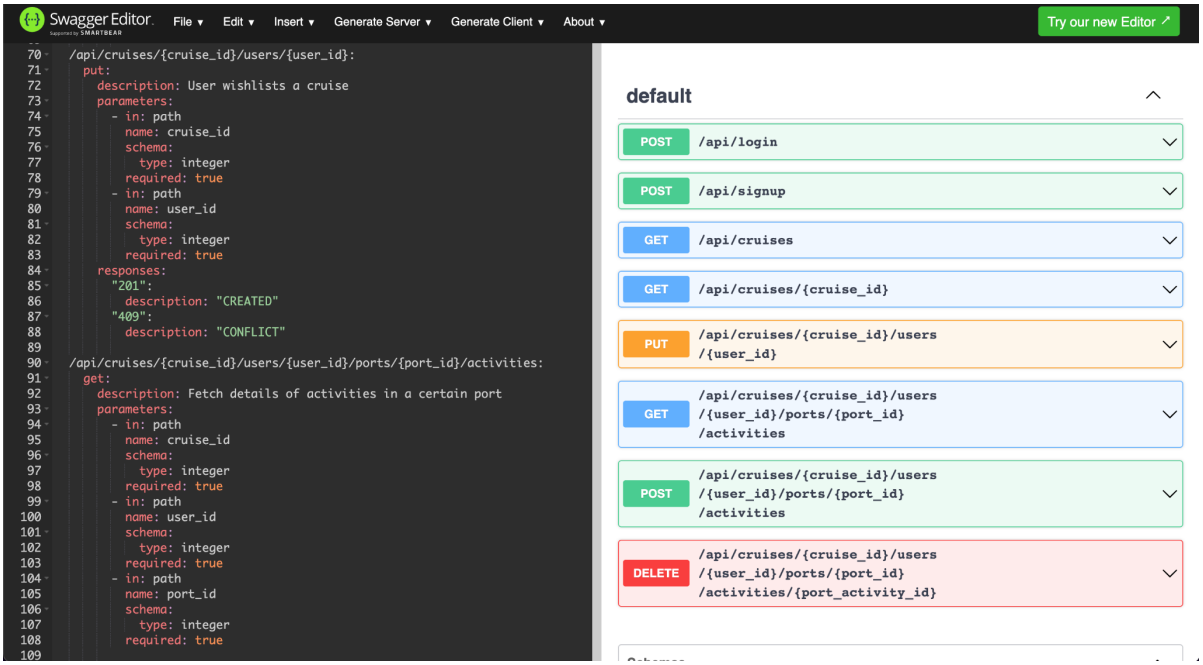
Crearea unui cont nou și autentificarea au fost realizate printr-o metoda de tip POST, deoarece datele de tip nume de utilizator și parola sunt considerate date sensibile. Pentru a asigura un grad ridicat de securitate, acestea trebuie trimise în corpul mesajului, nu ca parametrii ai URL-ului. Metoda POST a mai fost folosită pentru a adauga o activitate într-un port, în detrimentul metodei PUT, întrucât un utilizator trebuie să poată adaugă aceeași activitate de mai multe ori în cadrul unei opriri.

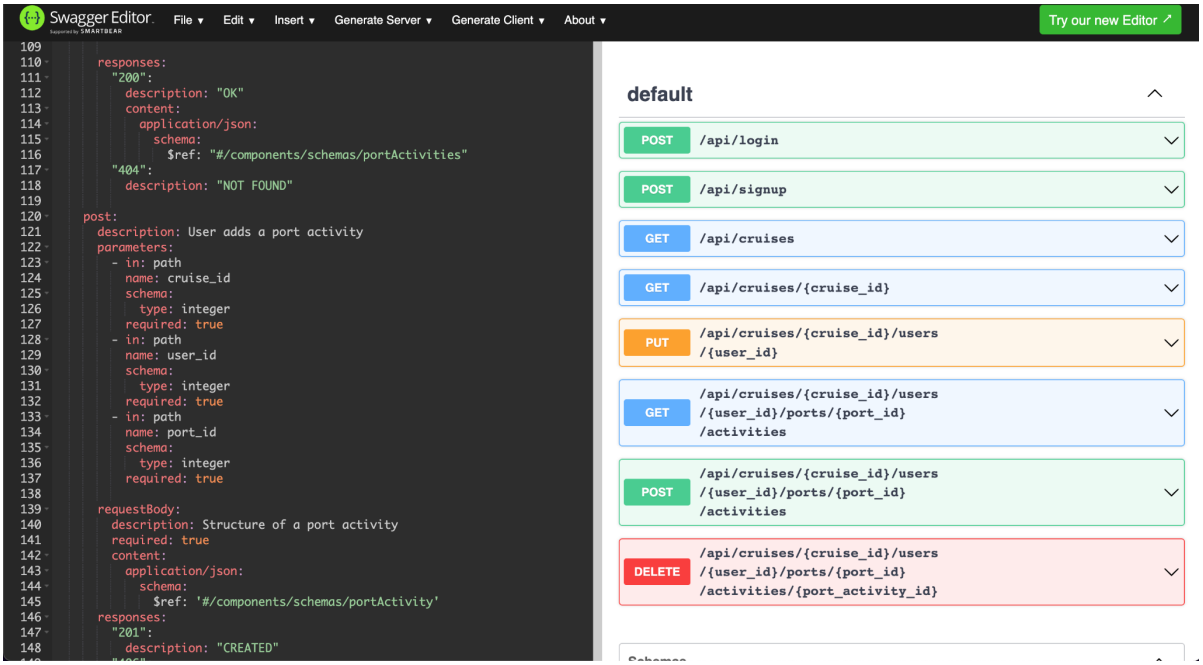
Pentru salvarea unei croaziere la preferințe de către un utilizator, s-a folosit metoda PUT, deoarece aceasta acțiune va avea același efect, indiferent de câte ori este realizată.

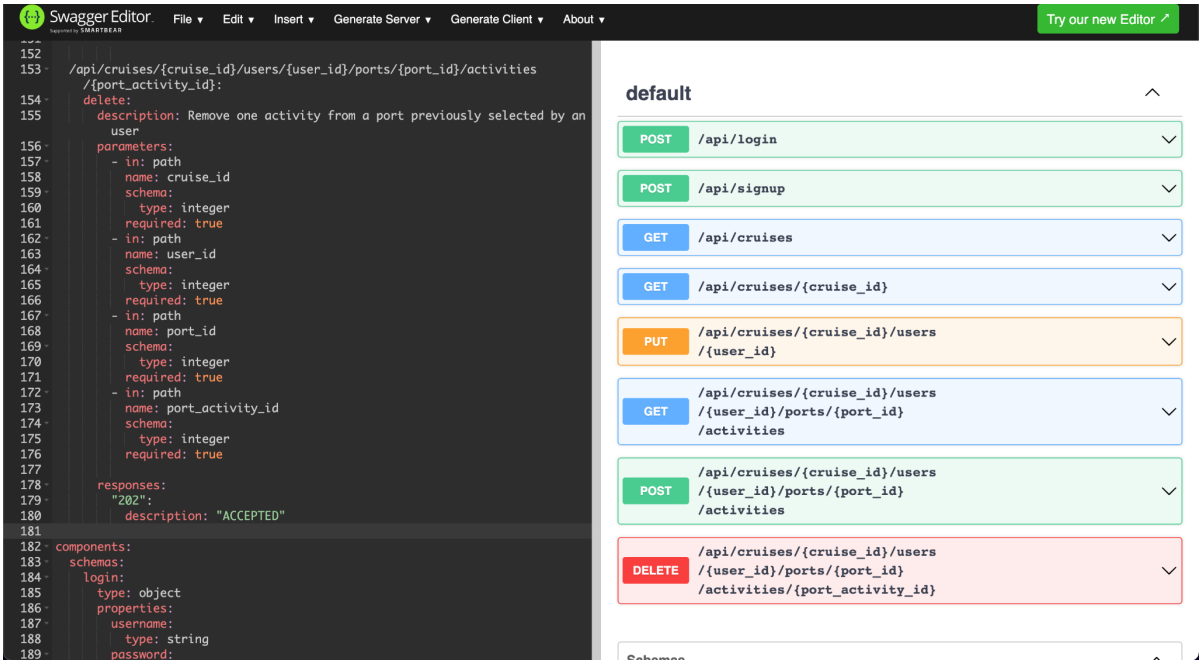
Metoda DELETE este utilizată pentru a șterge o croazieră din lista de preferințe.

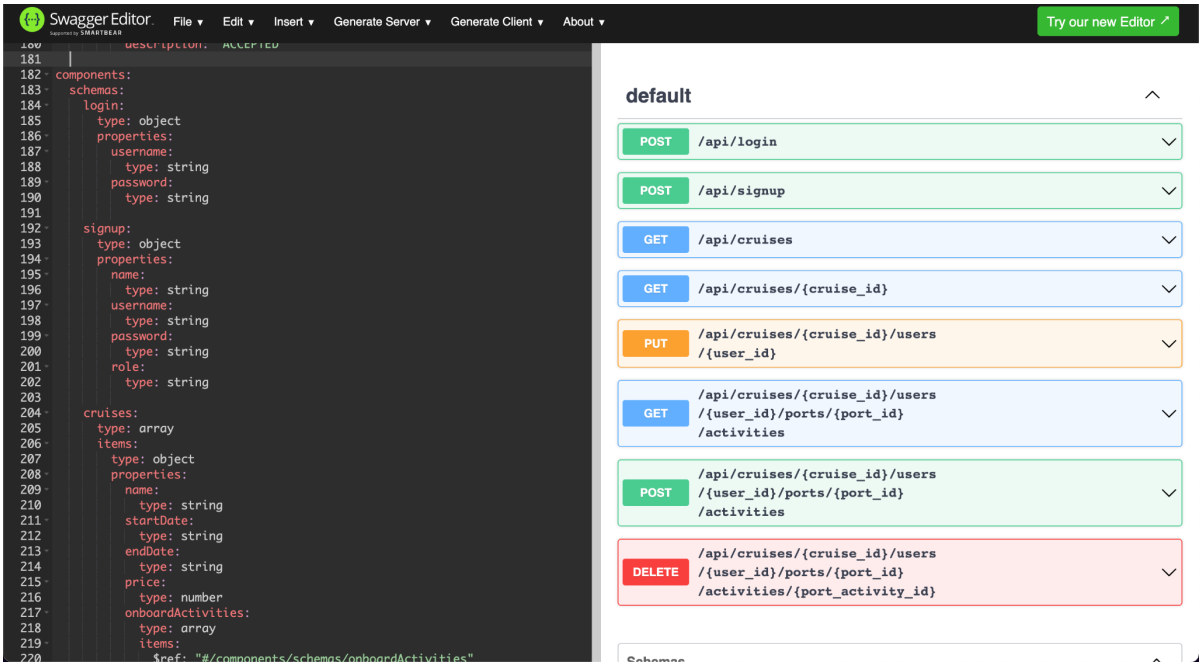


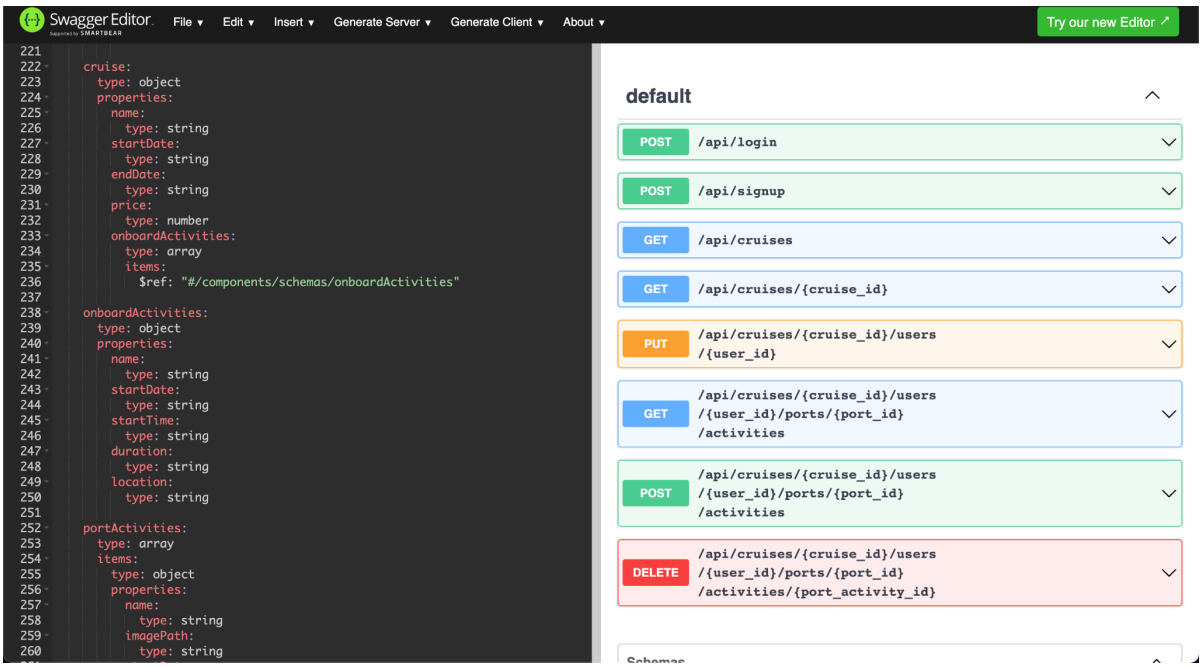

Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
About
Try our new Editor
/api/cruises:
get:
description: Fetch details of all cruises
responses:
"200":
description: "OK"
content:
application/json:
schema:
$ref: "#/components/schemas/cruises"
/api/cruises/{cruise_id}:
get:
description: Fetch details of a cruise
parameters:
- in: path
name: cruise_id
schema:
type: integer
required: true
responses:
"200":
description: "OK"
content:
application/json:
schema:
$ref: "#/components/schemas/cruise"
"404":
description: "NOT FOUND"
/api/cruises/{cruise_id}/users/{user_id}:
put:
description: User wishlists a cruise
parameters:
- in: path
name: cruise_id
schema:
type: integer
default
POST /api/login
POST /api/signup
GET /api/cruises
GET /api/cruises/{cruise_id}
PUT /api/cruises/{cruise_id}/users/{user_id}
GET /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
POST /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
DELETE /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities/{port_activity_id}

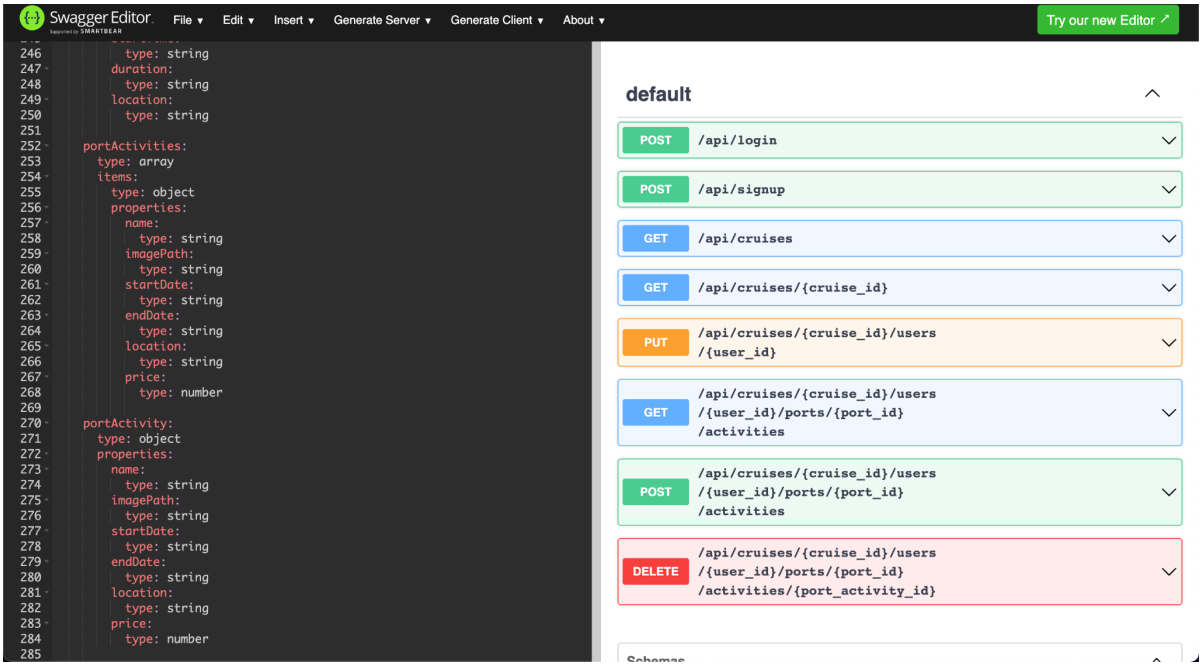

Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
About
Try our new Editor
/api/cruises/{cruise_id}/users/{user_id}:
put:
description: User wishlists a cruise
parameters:
- in: path
name: cruise_id
schema:
type: integer
required: true
- in: path
name: user_id
schema:
type: integer
required: true
responses:
"201":
description: "CREATED"
"409":
description: "CONFLICT"
/api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities:
get:
description: Fetch details of activities in a certain port
parameters:
- in: path
name: cruise_id
schema:
type: integer
required: true
- in: path
name: user_id
schema:
type: integer
required: true
- in: path
name: port_id
schema:
type: integer
required: true
default
POST /api/login
POST /api/signup
GET /api/cruises
GET /api/cruises/{cruise_id}
PUT /api/cruises/{cruise_id}/users/{user_id}
GET /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
POST /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
DELETE /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities/{port_activity_id}


Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
About
Try our new Editor
responses:
"200":
description: "OK"
content:
application/json:
schema:
$ref: "#/components/schemas/portActivities"
"404":
description: "NOT FOUND"
post:
description: User adds a port activity
parameters:
- in: path
name: cruise_id
schema:
type: integer
required: true
- in: path
name: user_id
schema:
type: integer
required: true
- in: path
name: port_id
schema:
type: integer
required: true
requestBody:
description: Structure of a port activity
required: true
content:
application/json:
schema:
$ref: '#/components/schemas/portActivity'
responses:
"201":
description: "CREATED"
default
POST /api/login
POST /api/signup
GET /api/cruises
GET /api/cruises/{cruise_id}
PUT /api/cruises/{cruise_id}/users/{user_id}
GET /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
POST /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
DELETE /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities/{port_activity_id}


Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
About
Try our new Editor
/api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities/{port_activity_id}:
delete:
description: Remove one activity from a port previously selected by an user
parameters:
- in: path
name: cruise_id
schema:
type: integer
required: true
- in: path
name: user_id
schema:
type: integer
required: true
- in: path
name: port_id
schema:
type: integer
required: true
- in: path
name: port_activity_id
schema:
type: integer
required: true
responses:
"202":
description: "ACCEPTED"
components:
schemas:
login:
type: object
properties:
username:
type: string
password:
default
POST /api/login
POST /api/signup
GET /api/cruises
GET /api/cruises/{cruise_id}
PUT /api/cruises/{cruise_id}/users/{user_id}
GET /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
POST /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
DELETE /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities/{port_activity_id}


Swagger Editor
File
Edit
Insert
Generate Server
Generate Client
About
Try our new Editor
description: ACCEPTED
components:
schemas:
login:
type: object
properties:
username:
type: string
password:
type: string
signup:
type: object
properties:
name:
type: string
username:
type: string
password:
type: string
role:
type: string
cruises:
type: array
items:
type: object
properties:
name:
type: string
startDate:
type: string
endDate:
type: string
price:
type: number
onboardActivities:
type: array
items:
$ref: "#/components/schemas/onboardActivities"
default
POST /api/login
POST /api/signup
GET /api/cruises
GET /api/cruises/{cruise_id}
PUT /api/cruises/{cruise_id}/users/{user_id}
GET /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
POST /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities
DELETE /api/cruises/{cruise_id}/users/{user_id}/ports/{port_id}/activities/{port_activity_id}
Schemas





GoogleMapsAPI este un serviciu Google ce facilitează :

-Specificarea coordonatele lat/lng pentru a centra harta.

-Afișarea unuia sau mai multor seturi de marcatori pe hartă.

-Aplicarea stilurilor personalizate pe o hartă.

-Afișarea unor diferite tipuri de hărți.

-Desenarea căilor personalizabile între punctele de pe hartă.

-Definirea ferestrei prin specificarea locațiilor vizibile.

5 Concluzii

Planificarea unei croaziere se poate dovedi a fi un efort considerabil, așadar arhitectura unei astfel de aplicații trebuind să asigure scalabilitate, eficiență și un procent ridicat de uptime. OceanCross întrunește necesitățile unui potențial client, oferind o interfață intuitivă și accesibilă.

6 Anexa

Link către Swagger Editor: https://editor.swagger.io/