UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA
FACULTATEA DE MATEMATICĂ ŞI INFORMATICĂ
SPECIALIZAREA INFORMATICĂ

LUCRARE DE LICENŢĂ

Crearea si Folosirea unui API

Conducător ştiinţific
Dr. Conf. Suciu Mihai

Absolvent
Belea Radu

2023

ABSTRACT

Abstract: un rezumat în limba engleză cu prezentarea, pe scurt, a conținutului pe capitole, punând accent pe contribuțiile proprii și originalitate

Cuprins

Capitolul 1

Introducere

Introducere: Application Programming Interfaces (API) sunt componente esențiale în dezvoltarea de software din zilele noastre. Ele permit comunicarea între mai multe sisteme software și facilitează dezvoltarea de servicii și aplicații complexe. API-urile, după cum le spune și numele, reprezintă interfețe standard care maschează complexitatea codului, ceea ce permite folosirea funcționalităților fără a ști modul în care au fost implementate.

Pe măsură ce companiile au început să folosească serviciile de tip cloud, multe API-uri au fost create pentru a ajuta programatorii. Din acest motiv, trebuie asigurat faptul că un API este ușor de folosit și eficient.

În această lucrare, vom explora modul în care API-urile sunt create și folosite, tipurile diferite de API-uri și modurile în care sunt folosite. Vom enumera de asemenea și provocările de care avem parte în design-ul și dezvoltarea lor. Documentația, testarea și versionarea API-urilor este de asemenea importantă, și vom prezenta și niște exemple de modele de API-uri. Vom intra în detaliu despre partea practică, o aplicație web care explorează folosirea de API-uri publice, precum și crearea unui API nou.

Scopul final al lucrării este de a prezenta modul în care API-urile sunt create și folosite și de a oferi o imagine de ansamblu asupra tehnicilor ”best practice” adoptate în dezvoltarea aplicațiilor folosind API-uri.

Capitolul 2

Partea teoretica

2.1 API Design

Joshua Bloch [Jos] mentioneaza in prezentarea sa ca un API trebuie sa fie simplu, consistent si flexibil pentru a fi considerat un design bun. El enumera de asemenea si niste sfaturi si tehnici standard folosite in acest proces, precum si principii importante in design-ul de API, cum ar fi denumiri clare, consistenta in modelele folosite, lizibilitate, documentatie, gestionarea eficienta a erorilor si versionarea.

Importanta conventiilor de denumire revine din folosirea numelor descriptive pentru clase, metode si variabile. Bloch sugereaza evitare abrevierilor si acronimelor care ar putea fi neclare, atat pentru utilizatori cat si pentru programatori care vor lucra la API in viitor. Astfel, denumirile clare si consistente sunt importante pentru utilizabilitatea API-urilor deoarece faciliteaza folosirea mai usoara a API-urilor.

Un principiu la fel de important este consistenta in modelele de design folosite. Pe langa conventiile de denumire, un API bun trebuie sa pastreze consistenta intre toate elementele sale de design. Aceasta usureaza scalabilitatea si mentenanta si permite programatorilor sa modifice si sa faca actualizari fara a strica elemente deja existente in aplicatie.

Pentru a ne asigura ca un API este usor de inteles si folosit, documentatia trebuie sa fie lizibila si destul de detaliata intrucat un programator sa poata reduca dificultatea de intelegere pentru utilizatori noi si sa reduca situatiile in care pot aparea erori.

Un alt principiu pe care Bloch pune accentul este gestionarea eficienta a erorilor. Dezvoltatorii unui API robust trebuie sa anticipeze si sa gestioneze potentialele erori care pot aparea intr-un mod clar si informativ. Acest fapt poate fi atins prin mesaje clare de eroare, ceea ce duce la API-uri folositoare si robuste.

Versionarea unui API este un principiu de baza pentru mentinerea compatibilitatii dintre API si aplicatiile in care este folosit. Astfel, actualizari ale API-ului pot fi

facute fara a strica functionalitati deja existente.

2.2 Implementare de API

Implementarea de API-uri, dupa cum este prezentata de Joshua Bloch, implica, de asemenea, mai multe principii cheie. Acestea sunt simplitatea, flexibilitatea, performanta, securitatea si utilizabilitatea.

Simplitatea este unul din cele mai importante principii de implementare a API-urilor. Un API simplu foloseste interfete clare, evita complexitatea si minimizeaza dependintele dintre componente, ceea ce duce la un numar crescut de utilizatori.

Flexibilitatea API-urilor revine din dezvoltarea unui API care este usor de modificat, adaptabil si extensibil. Un programator poate adauga functionalitati noi fara a face modificari majore.

Boch recomanda dezvoltarea API-urilor intr-un mod eficient si performant pentru a minimiza timpii de asteptare si a imbunatatii experienta utilizatorilor. Acest lucru poate fi facut folosind structure de date si algoritmi adecvati si aplicarea tehnicilor de optimizare asupra transferului de date si uzului de resurse.

Un punct major de risc al API-urilor este securitatea. Implementarea API-urilor ar trebui sa includa mecanisme de autentificare securizata precum si protejarea impotriva atacurilor cibernetice comune, cum ar fi SQL injection si cross-site scripting.

In final, utilizabilitatea API-urilor consta in implementarea API-urilor intr-un mod usor de inteles si utilizat, prin documentatie si interfete intuitive si folosirea mesajelor de eroare informative.

2.3 Tipuri de API

Exista mai multe tipuri de API-uri. Vom enumera cateva in acest capitol, urmand ca apoi sa intram in mai multe detalii despre API-ul REST.

2.3.1 REST API

Representaional State Transfer (REST) este unul dintre cele mai populare tipuri de API. Este construit pe baza cererilor de tip HTTP, prin care poate accesa si modifica resurse precum servicii web sau informatii din baze de date. Sunt ideale pentru a dezvolta aplicatii mobile sau web.

2.3.2 SOAP API

API-urile Simple Object Acces Protocol (SOAP) folosesc fisiere XML pentru a

transfera date intr-un format web. Sunt construite pe baza unor standarde precum Web Services Description Language (WSDL) sau Universal Description, Discovery and Integration (UDDI), ceea ce ofera functionalitati si functii de securitate mai complexe. Sunt folosite pentru aplicatii de tip enterprise.

2.3.3 GraphQL API

API-urile GraphQL sunt un tip mai nou de API bazat pe limbaj query pentru a crea cereri directe catre un server sau alt furnizor de date. Datorita eficientei acestui proces, sunt ideale pentru aplicatii care necesita accesul datelor rapid si eficient, dar este mai greu de implementat decat celelalte tipuri de API.

2.3.4 WebSocket

Websocket-urile permit comunicarea directa dintre un client si un server, folosind o conexiune persistenta intre cele doua entitati. In acest fel, se evita nevoia repetarii cererilor si raspunsurilor. Websocket-urile sunt folosite in principal pentru aplicatii care necesita sincronizarea datelor in timp real, precum aplicatii de chat sau jocuri online. De asemenea, pot fi folosite pentru a implementa notificari sau analiza real-time.

Capitolul 3

REST API

Datorita simplitatii sale, API-ul REST a devenit unul dintre cele mai populare tipuri de API folosite in aplicatiile moderne. REST reprezinta o metoda standard de a integra servicii web intr-o aplicatie si permite clientilor sa interactioneze cu resurse de pe server prin operatii bine-definite.

Prin publicatia sa din 2016, Vijay Surwase[Sur16] evidentiaza importanta modelarii unui API REST. Structura si comportamentul API-ului pot varia in functie de modul in care este definit si ajuta in alcatuirea documentatiei si a comunicarii functionalitatii aplicatiei.

Lucrarea stiintifica adreseaza de asemenea provocarile care pot aparea in dezvoltarea API-urilor RESTful, cum ar fi definirea resurselor si relatiei cu restul aplicatiei, gestionarea starilor si a tranzitiilor si asigurarea compatibilitatii. Aceste provocari pot creste atat complexitatea API-ului, precum si sansele de a aparea erori. Aceste fapte pot duce la dificultati in mentinerea si imbunatatirea API-ului si in comunicarea functionalitatii cu clientul sau alti programatori.

Pentru a putea depasi aceste provocari, Surwase enumera limbaje de modelare care ofera modalitati pentru a dezvolta si documenta API-uri REST. Acestea sunt:

- RESTful API Modeling Language (RAML), un limbaj de modelare open-source folosit pentru a defini resurse, parametri, scheme de cereri si raspunsuri si exemple. Este simplu, accesibil si usor de folosit.

- OpenAPI Specification este cel mai popular limbaj de modelari pentru API-urile REST deoarece permite programatorilor sa defineasca structuri, scheme de cereri si raspunsuri sau metode de autentificare, sa genereze documentatii, librarii pentru clienti si sa integreze metode de testare.

- API Blueprint, un limbaj de specificare de nivel inalt bazat pe Markdown care permite definirea documentatiei menita utilizatorilor int-un mod lizibil si prietenos.

- JSON Schema, care este folositor in principal in definirea structurii si regulilor de validare pentru API-urile RESTful bazate pe JSON. Folosit in general impreuna cu OpenAPI Specification.

Fiecare limbaj vine cu avantajele si dezavantajele sale, pe care lucrarea le evidentiaza comparandu-le intre ele.

Nume	Sponsor	Format	Open Source	Generare cod client	Generare cod server
RAML	MuleSoft	YAML	Da	Limitata	Da
API Blueprint	Apiary	Markdown	Da	Nu	Da
OpenAPI	Reverb	JSON	Da	Da	Da

Tabela 3.1: Comparatie intre limbaje de modelare

Desi toate limbajele sunt de tip Open Source, doar OpenAPI ofera generare de cod si pentru client.

RAML si API Blueprint sunt limbaje accesibile si usor de integrat intr-o aplicatie si ofera un mod comod de a crea documentatii pentru API.

OpenAPI si RAML ofera o consola de acces care permite programatorilor sa interactioneze mai usor cu API-ul. De asemenea, sunt compatibile cu majoritatea limbajelor de programare.

OpenAPI, fiind cel mai popular limbaj, are o comunitate extinsa, ceea ce faciliteaza rezolvarea rapida a problemelor.

Capitolul 4

Concluzii

Concluzii ...

Bibliografie

[Jos] Joshua Bloch. How to Design a Good API and Why it Matters.

[Sur16] Vijay Surwase. Rest api modeling languages - a developer's perspective. *IJSTE - International Journal of Science Technology Engineering*, 2(10), 2016.