

Universiteti për Biznes dhe Teknologji

Master Shkenca Kompjuterike dhe Inxhinieri



Sistemi për Menaxhimin e Telefonave - SMT

Lënda: Inxhinieria Softuerike për Shkallëzueshmëri të Aplikacionit

Profesoreshë:

Dr. Sc. Ermira Daka

Punoi:

Elma Redzepi

Prishtinë, 2025

Sistemi për Menaxhimin e Telefonave - SMT

Elma Redzepi – er242575418@ubt-uni.net

University for Business and Technology, 10000 Prishtine, Kosovo

Abstrakti: Ky projekt synon të ofrojë një zgjidhje moderne dhe funksionale për menaxhimin e telefonave në një mjedis organizativ apo biznesor përmes zhvillimit të një sistemi të quajtur **SMT – Sistemi për Menaxhimin e Telefonave**. SMT është një aplikacion i ndërtuar mbi arkitekturën **Client–Server**, ku pjesa e backend-it është zhvilluar me **Slim PHP Framework**, ndërsa të dhënat ruhen në një **bazë të dhënash MySQL**. Pjesa frontend ndërvepron përmes API-ve RESTful të testuara me **Postman**, duke garantuar komunikim të sigurt dhe të qëndrueshëm midis përdoruesit dhe sistemit. Ky sistem mbështet të gjitha operacionet bazë **CRUD (Create, Read, Update, Delete)** për të menaxhuar të dhënat e telefonave si emri, marka, modeli, çmimi, sasia dhe gjendja. Struktura e bazës së të dhënave është ndërtuar në mënyrë të normalizuar për të mundësuar qasje efikase, integritet të të dhënave dhe zgjerueshmëri në të ardhmen. Për më tepër, përmes përdorimit të **phpMyAdmin** dhe **XAMPP**, sigurohet një ambient zhvillimi lokal stabil dhe i lehtë për menaxhim. Në projekt janë realizuar testime funksionale për të gjitha operacionet kryesore përmes Postman, duke konfirmuar saktësinë dhe integritetin e shërbimeve të ofruara nga API-të. Megjithatë testimet për shkallëzueshmëri (scalability) nuk janë përfshirë në mënyrë të plotë, struktura modulare e sistemit e bën atë të përgatitur për integritet të ardhshme dhe përpunim të sasive më të mëdha të të dhënave. SMT paraqet një shembull praktik të një aplikacioni të plotë me arkitekturë të ndarë dhe logjikë të qartë biznesore, i cili mund të shërbejë si model për sisteme të tjera të menaxhimit të produkteve.

PËRMBJATJA

LISTA FIGURAVE	4
1. HYRJE	5
2. DIZAJNI ARKITEKTURËS – (HIGH-LEVEL ARCHITECTURE)	6
3. DIZAJNI DATABAZES.....	8
4. IMPLEMENTIMI I APLIKACIONIT PËRMES POSTMAN-API BACK-end DHE FRONT-end	9
4.1. RESTful API – Backend.....	9
4.1.1. Paraqitja e të gjitha të dhënave	11
4.1.2. Paraqitja e një të dhëne të caktuar sipas ID-së	12
4.1.3. Shtimi i të dhënave.....	12
4.1.4. Modifikimi i të dhënave.....	14
4.1.5. Fshirja e të dhënave	15
4.2. RESTful API – Frontend.....	16
4.2.1. Paraqitja e të gjitha të dhënave	17
4.2.2. Kërkimi i të dhënave.....	18
4.2.3. Shtimi i të dhënave.....	19
4.3. 4. Modifikimi i të dhënave.....	20
4.2.5. Fshirja e të dhënave	21
5. CACHING, PERFORMANCE OPTIMIZATION, EVENT-DRIVEN ARCHITECTURE, FAULT TOLERANCE, HIGH AVAILABILITY, AND RESILIENCE	22
5.1. Caching and Performance Optimization	22
5.2. Fault Tolerance	23
5.3. High Availability and Resilience.....	23
6. SCALABILITY TESTS.....	24
6.1. Test Scripts në Postman (në tabin <i>Tests</i> për secilin request):.....	24
6.2. Tests Scripts ne Postman per shtimin e testeve:.....	25
6.3. Tests Scripts ne Postman per modifikimin e testeve:	28
6.4. Tests Scripts ne Postman per modifikimin e testeve:	30
7. PËRFUNDIMI	33
REFERENCAT	34

LISTA FIGURAVE

Figure 1. Dizajni Arkitekturës	7
Figure 2. Databaza	8
Figure 3. Paraqitja e Postman – Backend	10
Figure 4. Paraqitja e të gjitha të dhënave – Backend	11
Figure 5. Paraqitja e një të dhënë – Backend	12
Figure 6. Shtimi i të dhënave – Backend	13
Figure 7. Modifikimi i të dhënave – Backend.....	15
Figure 8. Fshirja e të dhënave – Backend	16
Figure 9. Paraqitja e të dhënave – Frontend	17
Figure 10. Paraqitja e të dhënave – Frontend	18
Figure 11. Kërkimi i të dhënave – Frontend.....	19
Figure 12. Shtimi i të dhënave -Frontend	20
Figure 13. Modifikimi i të dhënave – Frontend	21
Figure 14. Fshirja e të dhënave – Frontend	22
Figure 15. Paraqitja e të dhënave me GET në Scripts	24
Figure 16. Shtimi i të dhënave POST	25
Figure 17. Gjenerimi i Scripts me POST.....	26
Figure 18. Scripts POST Run API.....	26
Figure 19. Gjenerimi i 100 Scripts – POST.....	27
Figure 20. Pas gjenerimit të 100 Scripts - Databaza	27
Figure 21. Paraqitja pas gjenerimit të Scripts edhe në Front-end	28
Figure 22. Tests Scripts - Modifikimi i një të dhëne.....	28
Figure 23. Gjenerimi i Scripts	29
Figure 24. Kemi gjeneruar 1 të dhënë me PUT	29
Figure 25. Gjenerimi i Scripts me PUT	30
Figure 26. Fshirja e në të dhëne.....	30
Figure 27. Gjenerimi i Scripts - DELETE	31
Figure 28. Gjenerimi i 10 Scripts me DELETE	31
Figure 29. Gjenerimi i të dhënave me DELETE.....	32

1. HYRJE

Në epokën moderne të dixhitalizimit, menaxhimi efikas i pajisjeve teknologjike përbën një nevojë thelbësore për shumë biznese dhe institucione. Telefonat, si një prej mjeteve më të përdorura në komunikim dhe operacione të përditshme, kërkojnë një sistem të besueshëm për regjistrim, kontroll dhe përditësim të të dhënave. Për këtë arsye, u ndërtua projekti “**Sistemi për Menaxhimin e Telefonave – SMT**”, me qëllim të ofrojë një platformë të thjeshtë, por funksionale për trajtimin e informacionit në lidhje me telefonat.

SMT është një sistem me arkitekturë të ndarë, ku pjesa backend është ndërtuar duke përdorur **Slim PHP Framework**, i njohur për lehtësinë e tij dhe përkrahjen ndaj zhvillimit të API-ve RESTful. Këto API lejojnë komunikimin me pjesën frontend ose përdoruesin përmes kërkesave **GET, POST, PUT**, dhe **DELETE**. Si ambient zhvillimi është përdorur **XAMPP**, që integron serverin Apache dhe sistemin e menaxhimit të databazës MySQL për një përvojë testimi dhe zhvillimi sa më të mirë. Struktura e bazës së të dhënave përfshin tabela të dizajnuara për të mbështetur funksionalitetin e menaxhimit të telefonave, duke ruajtur fushat si: *id, emri, marka, modeli, qmimi, sasia, dhe gjendja*. Në aspektin e testimeve, projekti është verifikuar funksionalisht përmes **Postman**, ku janë testuar të gjitha endpoint-et e API-së për të garantuar se çdo funksion i implementuar punon saktë. Për shembull, janë testuar shtimi i një telefoni të ri, përditësimi i të dhënave të një telefoni ekzistues, fshirja e telefonit, dhe marrja e listës së të gjithë telefonave. Këto testime demonstrojnë integrimin dhe qëndrueshmërinë e ndërveprimit mes komponentëve të sistemit. Qëllimi i këtij projekti nuk është vetëm të ofrojë një zgjidhje funksionale për menaxhimin e telefonave, por gjithashtu të demonstrojë përdorimin praktik të koncepteve të **Inxhinierisë së Softuerit**, përfshirë: analiza e kërkesave, dizajni i sistemit, implementimi i kodit, testimi dhe dokumentimi. SMT është ndërtuar në mënyrë të tillë që të jetë **lehtësisht i zgjerueshëm**, duke mundësuar në të ardhmen shtimin e funksionaliteteve si: kërkimi dinamik, kategorizimi i telefonave sipas markës, raportimi statistikor, apo autentifikimi i përdoruesve.

Ky projekt përfaqëson një hap të rëndësishëm drejt ndërtimit të sistemeve praktike që adresojnë nevoja konkrete dhe është i përshtatshëm për t'u përdorur si model për zgjidhje të tjera të ngjashme në fushën e menaxhimit të produkteve apo inventarit në organizata të ndryshme.

2. DIZAJNI ARKITEKTURËS – (HIGH-LEVEL ARCHITECTURE)

Dizajnimi i arkitekturës së këtij sistemi përfaqëson një qasje të strukturuar dhe të modularizuar, e cila e ndan sistemin në tri shtresa thelbësore:

- Ndërfaqja e përdoruesit (Front-End),
- Shtresa e logjikës së aplikacionit dhe shërbimeve (Back-End), si dhe
- Shtresa e ruajtjes së të dhënave (Database).

Kjo ndarje tre-shtresore është themelore për ndërtimin e aplikacioneve moderne, pasi garanton ndarje të përgjegjëse, mirëmbajtje më të thjeshtë, testim të pavarur të komponentëve dhe mundësi për zgjerim në të ardhmen pa ndikuar negativisht në pjesët tjera të sistemit.

Në pikënisje qëndron **përdoruesi**, i cili është subjekti aktiv i ndërveprimit me sistemin. Ai ndërvepron përmes ndërfaqes së ndërtuar me **Vue.js**, një framework progresiv për ndërtimin e ndërfaqeve të përdoruesit (SPA – Single Page Application), që siguron ngarkesë të shpejtë, përditësime dinamike të përmbajtjes, dhe përdorim të shtrirë të komponentëve të ripërdorshëm. Përmes formave, komandave dhe ndërfaqeve vizuale të Vue.js, përdoruesi nis kërkesa ndaj serverit në mënyrë asinkrone përmes **HTTP (AJAX/Fetch)**, duke ndarë qartë UI nga logjika serverike.

Kërkesat HTTP kapen nga një API e ndërtuar në **Slim PHP**, një mikro-framework që mbështet RESTful API, i cili është zgjedhur për lehtësinë, fleksibilitetin dhe performancën e tij. Slim vepron si ndërmjetës logjik dhe përpunues biznesi: i verifikon, i validon dhe i transformon të dhënat që vijnë nga Front-End-i dhe më pas merr vendime sipas rregullave të logjikës së brendshme të sistemit (si validimi i të dhënave të një pajisjeje, kontrolli i autentikimit, apo ndarja e të drejtave të aksesit për përdoruesit e ndryshëm). Slim gjithashtu orkestron komunikimin me bazën e të dhënave përmes pyetjeve **SQL**.

Database Management System-i i përdorur është **MySQL**, i cili është zgjedhur për stabilitetin, strukturën relacione dhe përputhjen me PHP. Të dhënat ruhen në mënyrë të normalizuar në tabela që përfaqësojnë entitete të ndryshme si: Telefonat_SMT, Përdoruesit, Garancionet, SpecifikatTeknike etj. Struktura relacione garanton koherencë, integritet referencial dhe performancë të lartë në kërkesa të ndërlikuara me shumë bashkime (JOINS). Operacionet e CRUD-it (Create, Read, Update, Delete) mbështeten përmes një layer abstraksioni të ndërtuar mbi Slim PHP.

Lidhjet mes komponentëve përshkohen nga protokolle të standardizuara si **HTTP** për komunikim ndërmjet klientit dhe serverit, dhe **SQL** për ndërveprim me bazën e të dhënave. Për më tepër, sistemi është i integruar me mekanizma të automatizimit të zhvillimit si **CI/CD (Continuous Integration / Continuous Deployment)**. Përmes këtyre mekanizmave, çdo ndryshim në kod (në front-end apo back-end) testohet automatikisht, ndërtohet dhe publikohet në ambientin e prodhimit, duke e bërë zhvillimin më efikas dhe me më pak rreziqe për defekte në versionet live. Ky dizajn modular dhe i ndarë mirë garanton përfitime të shumëfishta në aspektin teknik dhe operacional. Ai siguron **shkallëzueshmëri horizontale** dhe **vertikale** të sistemit, mbështet testimin e njësiteve në mënyrë të pavarur, mundëson integritet të lehta me API të jashtme, dhe ofron bazën për vendosjen e praktikave moderne si caching, rate-limiting, auditing, logging, monitoring në kohë reale dhe aplikimin e standardeve të sigurisë si JWT, OAuth2 ose HTTPS.

Arkitektura është projektuar në mënyrë që të mbështesë jo vetëm nevojat e tanishme, por edhe të parashikojë zgjerime të ardhshme në drejtim të funksionalitetit, ngarkesës apo bashkëveprimit me sisteme të tjera, duke siguruar kështu një bazë solide për rritje të qëndrueshme dhe evoluim teknologjik.

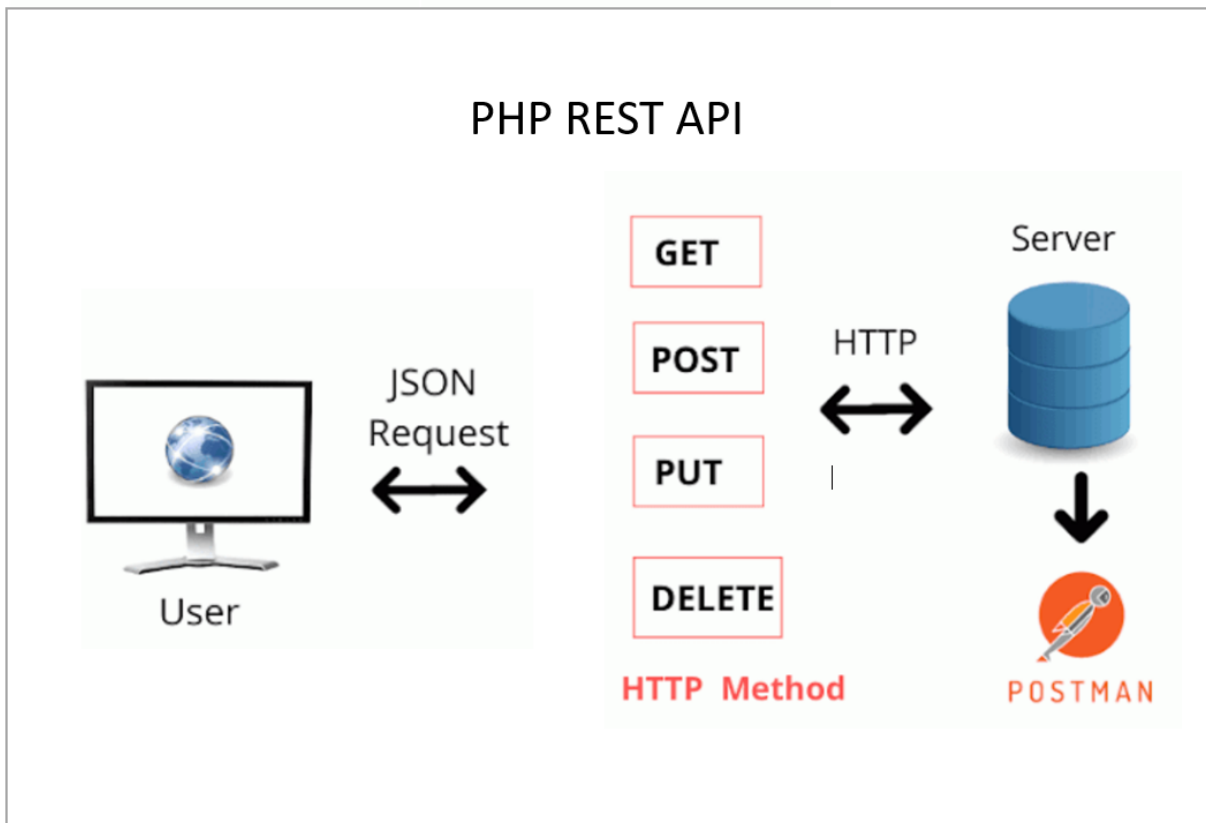
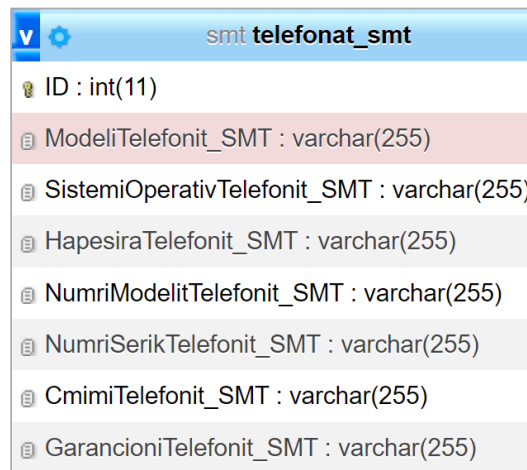


Figure 1. Dizajni Arkitekturės

3. DIZAJNI DATABAZES

Për të krijuar një sistem për menaxhimin e telefonave, hapi i parë është krijimi i një databaze MySQL që do të mbajë të dhënat për telefonat. Kjo mund të arrihet duke përdorur komandat e MySQL për krijimin e databazës dhe tabelës përkatëse. Në këtë rast, kemi krijuar databazën **SMT** dhe një tabelë të quajtur **Telefonat_SMT**, e cila do të ruajë informacionin e telefonave që do të menaxhohen. Kjo skemë është projektuar në mënyrë që të ofrojë mundësinë për ruajtjen e të dhënave të ndryshme që lidhen me telefonat, si modeli, sistemi operativ, hapësira, numri serik, çmimi dhe garancia, të cilat na shërbejnë për të kryer shërbime të ndryshme për menaxhimin e telefonave.



smt telefonat_smt	
ID	int(11)
ModeliTelefonit_SMT	varchar(255)
SistemiOperativTelefonit_SMT	varchar(255)
HapesiraTelefonit_SMT	varchar(255)
NumriModelitTelefonit_SMT	varchar(255)
NumriSerikTelefonit_SMT	varchar(255)
CmimiTelefonit_SMT	varchar(255)
GarancioniTelefonit_SMT	varchar(255)

Figure 2. Databaza

Shpjegimi i kolonave të tabelës **Telefonat_SMT**:

- **ID**: Kolona ID është çelësi kryesor (Primary Key, PK) i tabelës dhe është automatikisht në rritje (Auto Increment, AI). Ajo identifikon çdo telefon në mënyrë unike në tabelë.
- **ModeliTelefonit_SMT**: Ky kolon ruan modelin e telefonit. Përdoret për të identifikuar telefonin në bazë të markës dhe versionit të tij (p.sh., "iPhone 13").
- **SistemiOperativTelefonit_SMT**: Ky kolon ruan sistemin operativ që telefoni përdor (p.sh., "iOS", "Android").
- **HapesiraTelefonit_SMT**: Ky kolon ruan hapësirën e telefonit në gigabajt (GB). Përdoret për të specifikuar sa hapësirë ka telefoni për ruajtjen e të dhënave.
- **NumriModelitTelefonit_SMT**: Ky kolon ruan numrin e modelit të telefonit, që është i rëndësishëm për identifikimin e variacioneve të ndryshme të një modeli të caktuar.
- **NumriSerikTelefonit_SMT**: Ky kolon ruan numrin serik të telefonit, i cili është një identifikues unik për çdo telefon dhe është i dobishëm për gjurmimin dhe garancinë.
- **CmimiTelefonit_SMT**: Ky kolon ruan çmimin e telefonit në formatin decimal, përfshirë dy shifra pas presjes për saktësi të çmimit.
- **GarancioniTelefonit_SMT**: Ky kolon ruan kohëzgjatjen e garancionit të telefonit në vite, i cili mund të jetë një informacion i rëndësishëm për menaxhimin e shërbimeve dhe politikave të kthimit.

4. IMPLEMENTIMI I APLIKACIONIT PËRMES POSTMAN-API BACK-end DHE FRONT-end

4.1. RESTful API – Backend

Krijimi i një API me Slim PHP

Slim PHP është një framework i thjeshtë dhe i shpejtë për zhvillimin e aplikacioneve web dhe API-ve. Ai ofron mundësinë për të krijuar endpoint-e të ndryshme që mund të kryejnë operacione të ndryshme mbi të dhënat e databazës në mënyrë të shpejtë dhe të sigurt. Slim është një framework i lehtë, por shumë fleksibël, i cili mund të përdoret për të ndërtuar aplikacione me kërkesa të thjeshta dhe të avancuara. Ai është një opsion i shkëlqyer për zhvillimin e API-ve RESTful që përdorin metodën e komunikimit HTTP për të transferuar të dhëna midis klientëve dhe serverëve.

Në këtë projekt, API-ja do të përfshijë një seri endpoint-esh që mundësojnë operacione të ndryshme si:

1. Krijimi i një Telefoni të Ri

Ky endpoint do të mundësojë që përdoruesit të shtojnë telefonat e rinj në databazë. Përdoruesi do të dërgojë informacionin e telefonit (p.sh., modeli, sistemi operativ, hapësira, numri serik, etj.) në formën e një kërkesë HTTP POST, dhe API-ja do ta ruajë këtë informacion në tabelën Telefonat_SMT. Kjo mundëson që përdoruesit të shtojnë telefonë të rinj me informacion të plotë dhe të saktë.

2. Leximi i të Dhënave të Telefonëve

Ky endpoint mundëson që përdoruesit të lexojnë të dhënat e telefonëve që janë tashmë të ruajtura në databazë. Përdoruesi mund të kërkojë një listë të të gjitha telefonave që janë regjistruar në sistem, ose mund të kërkojë të dhëna specifike për një telefon të caktuar duke përdorur ID-në e tij. Ky operacion është i rëndësishëm sepse mundëson shikimin e informacionit në kohë reale dhe siguron që përdoruesi të ketë mundësi të monitorojë dhe menaxhojë telefonat që janë regjistruar në sistem.

3. Përditësimi i të Dhënave të Telefonëve

Përdoruesit mund të kenë nevojë të përditësojnë të dhënat e telefonëve të regjistruar. Kjo mund të ndodhë për arsye të ndryshme, si përditësimi i çmimit të telefonit, ndryshimi i modelit të telefonit, ose ndonjë informacion tjetër që mund të ndryshojë me kalimin e kohës (p.sh., përditësimi i informacionit të garancisë). Ky endpoint mundëson përditësimin e të dhënave ekzistuese në databazë, dhe gjithashtu siguron që të dhënat të mbeten të sakta dhe të freskëta.

4. Fshirja e një Telefoni nga Databaza

Ky endpoint mundëson fshirjen e një telefoni të caktuar nga databaza, nëse nuk është më i nevojshëm ose për shkak të ndonjë arsye tjetër që lidhet me menaxhimin e telefonave në sistem. Përdoruesi mund të kërkojë fshirjen e telefonit nëpërmjet ID-së së tij, dhe API-ja do të sigurojë që të dhënat përkatëse të fshihen nga tabelat përkatëse të databazës.

Funksionalitetet e API-së dhe Interaksioni me Databazën

Për të siguruar që API-ja funksionon në mënyrë të përshtatshme, është e rëndësishme që ajo të jetë e ndërtuar në një mënyrë që mundëson operacione të sakta dhe efikase mbi të dhënat e databazës. Kjo do të

thotë që çdo kërkesë e dërguar në API duhet të përpunojë informacionin në mënyrë të saktë dhe të sigurojë që nuk ka gabime në ruajtjen e të dhënave.

Siguria dhe Autentifikimi i API-së

Në krijimin e një API-je, është shumë e rëndësishme të merren parasysh aspekte të sigurisë, përfshirë autentifikimin dhe autorizimin. Ky aspekt është veçanërisht i rëndësishëm për të siguruar që vetëm përdoruesit e autorizuar të mund të bëjnë ndryshime në të dhënat e telefonave. Përdorimi i mekanizmave të autentifikimit si JSON Web Tokens (JWT) ose OAuth mund të ndihmojë në sigurinë e API-së dhe në mbrojtjen e të dhënave sensitive. Përveç kësaj, është e rëndësishme të merret parasysh përdorimi i metodave të sigurisë si HTTPS për të siguruar që të dhënat që kalojnë ndërmjet klientit dhe serverit të jenë të mbrojtura nga ndonjë sulm i mundshëm, si p.sh. ndërhyrja e të dhënave (man-in-the-middle attacks).

Pas krijimit të databazës dhe tabelës, hapi i natyrshëm tjetër është krijimi i një API-je për të mundësuar kryerjen e operacioneve të ndryshme mbi të dhënat që ruhen në tabelën **Telefonat_SMT**. Ky API është një ndërfaqe që lejon komunikimin midis aplikacionit frontend dhe databazës përmes protokolleve të thjeshta të HTTP. API-ja mundëson që të dhënat e telefonave të jenë të aksesueshme dhe të menaxhueshme nga përdoruesit, si dhe mundëson operacione të ndryshme si: **Create, Read, Update** dhe **Delete** - të njohura ndryshe si **CRUD**, e të dhënave që ndodhen në databazën tonë. Postman është një aplikacion i cili përdoret për back-end. Ky aplikacion na ofron mundësi shërbyese shumë të mira i cili na ndimon të paraqesim të dhënat që kemi në databazë, nëse i kërkojmë të gjitha na paraqet të gjitha, nëse kërkojmë një të dhënë na shfaq një të dhënë, na shërben për të shtuar të dhëna, për të modifikuar të dhëna si dhe për të fshirë të dhënat që dëshirojmë.

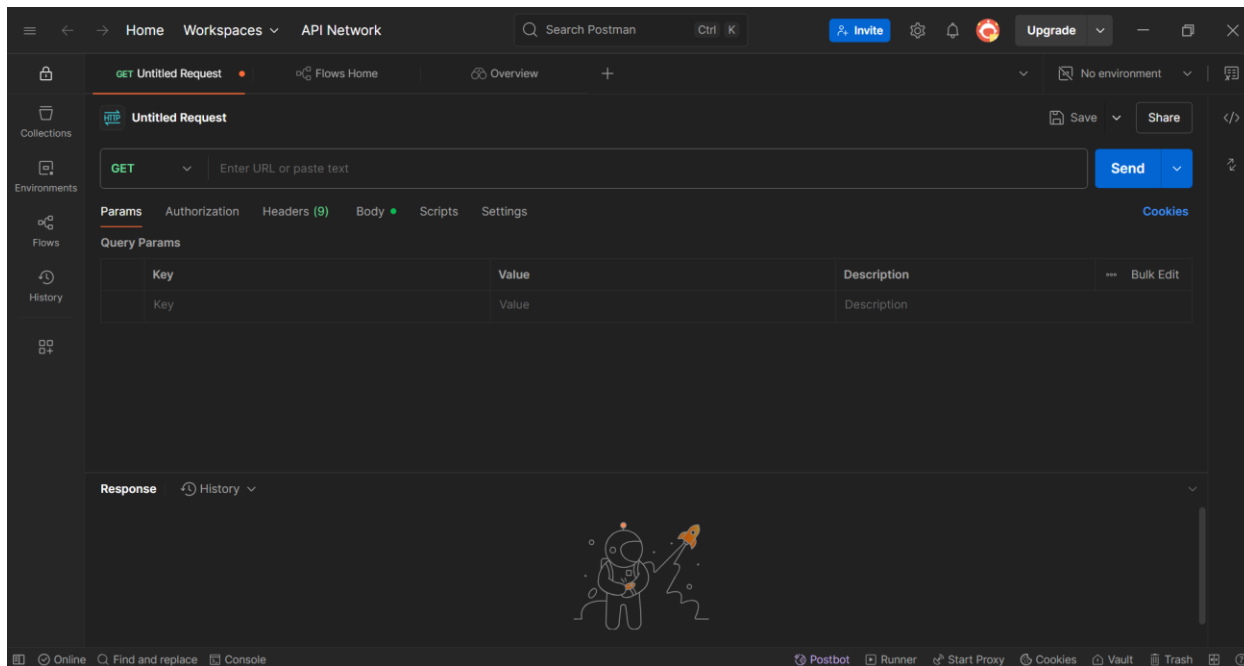


Figure 3. Paraqitja e Postman – Backend

4.1.1. Paraqitja e të gjitha të dhënave

Në këtë hap përdorim metodën **GET**, e cila është e dizajnuar për të **lexuar të dhënat** nga serveri.

Në aplikacionin **Postman**, pasi hapim një dritare të re për testim:

- Zgjedhim metodën GET nga menyja e metodave HTTP.
- Tek fusha e URL-së shkruajmë endpoint-in e krijuar në backend, që për shembull mund të jetë:

http://localhost/sliampppp/public/telefonat

- Me këtë kërkesë, ne kërkojmë nga API-ja që të **na kthejë të gjitha të dhënat** ekzistuese në tabelën Telefonat_SMT.
- Pasi të klikojmë **Send**, Postman komunikon me serverin dhe na paraqet në seksionin e përgjigjes të gjitha rreshtat ekzistues në formatin **JSON**, që përfshin të dhëna të tilla si:
 - ID
 - Modeli i telefonit
 - Sistemi operativ
 - Hapësira
 - Numri i modelit
 - Numri serik
 - Çmimi
 - Garancioni
- Kjo na ndihmon të sigurohemi që API-ja është funksionale dhe që të dhënat po lexohen saktësisht nga databaza.

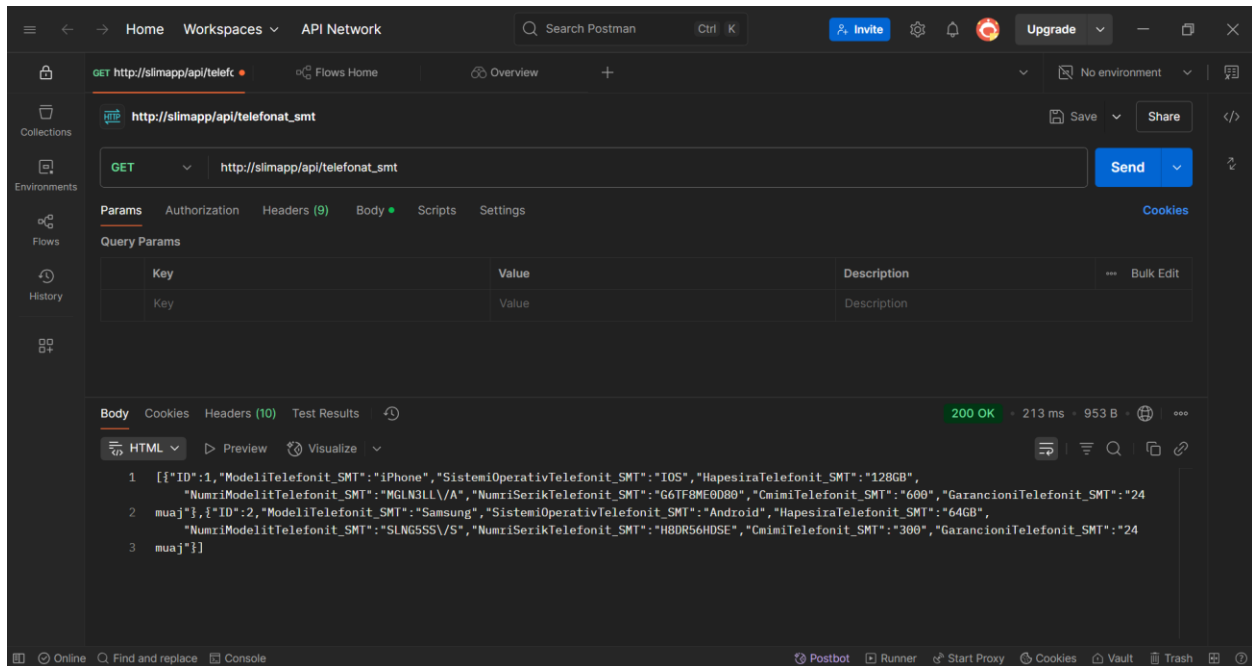


Figure 4. Paraqitja e të gjitha të dhënave – Backend

4.1.2. Paraqitja e një të dhëne të caktuar sipas ID-së

Përdorimi i metodës **GET** në këtë rast shërben për të marrë **vetëm një të dhënë specifike**, bazuar në **ID-në unike** të saj.

- Në Postman, zgjedhim metodën GET dhe në URL vendosim endpoint-in së bashku me ID-në e të dhënës që dëshirojmë ta shohim, si p.sh.:

<http://localhost/sliampppp/public/telefonat/3>

- Kjo kërkesë i thotë API-së të kërkojë në databazë për rreshtin që ka **ID = 3**, dhe të na e kthejë atë në përgjigje.
- Rezultati paraqitet në format JSON dhe përmban të gjitha fushat e plotësuara për atë telefon specifik.
- Ky funksionalitet është veçanërisht i dobishëm kur duam të konsultohemi vetëm me një produkt pa pasur nevojë të shkarkojmë të gjitha të dhënat.

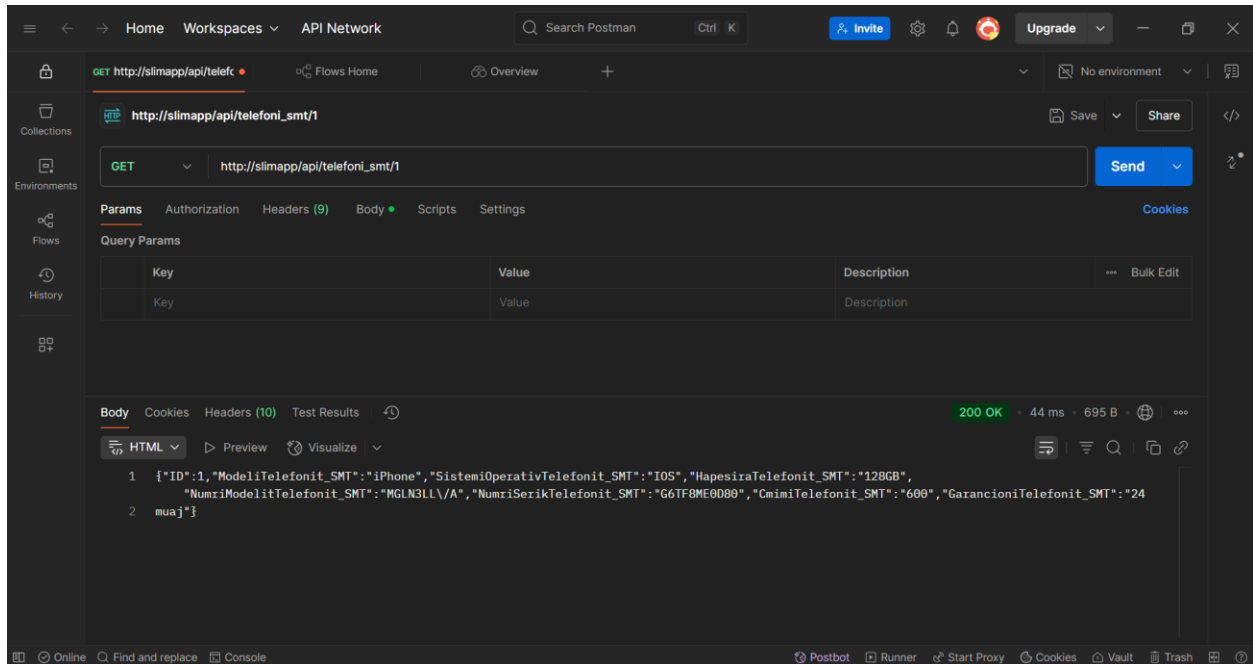


Figure 5. Paraqitja e një të dhëne – Backend

4.1.3. Shtimi i të dhënave

Ky hap përfshin **krijimin e të dhënave të reja në databazë**, përmes metodës HTTP **POST**.

- Në Postman, zgjedhim metodën POST.
- Vendosim URL-në për endpoint-in që pranon shtimin e të dhënave, si p.sh.:

<http://localhost/sliampppp/public/telefonat/add>

- Më pas, klikojmë në skedën **Body**, zgjedhim opsionin **raw**, dhe si format zgjedhim **JSON**.
- Në këtë seksion shtojmë një strukturë JSON që përmban të dhënat që dëshirojmë të shtojmë. P.sh.:

```
{  
  
  "ModeliTelefonit_SMT": "Samsung Galaxy S23",  
  
  "SistemiOperativTelefonit_SMT": "Android",  
  
  "HapesiraTelefonit_SMT": "256GB",  
  
  "NumriModelitTelefonit_SMT": "SM-S911B",  
  
  "NumriSerikTelefonit_SMT": "R5CT1234567",  
  
  "CmimiTelefonit_SMT": "850",  
  
  "GarancioniTelefonit_SMT": "24 muaj"  
}
```

Pas klikimit të **Send**, nëse gjithçka është konfiguruar siç duhet, do të marrim një përgjigje që konfirmon se të dhënat janë shtuar me sukses në databazë.

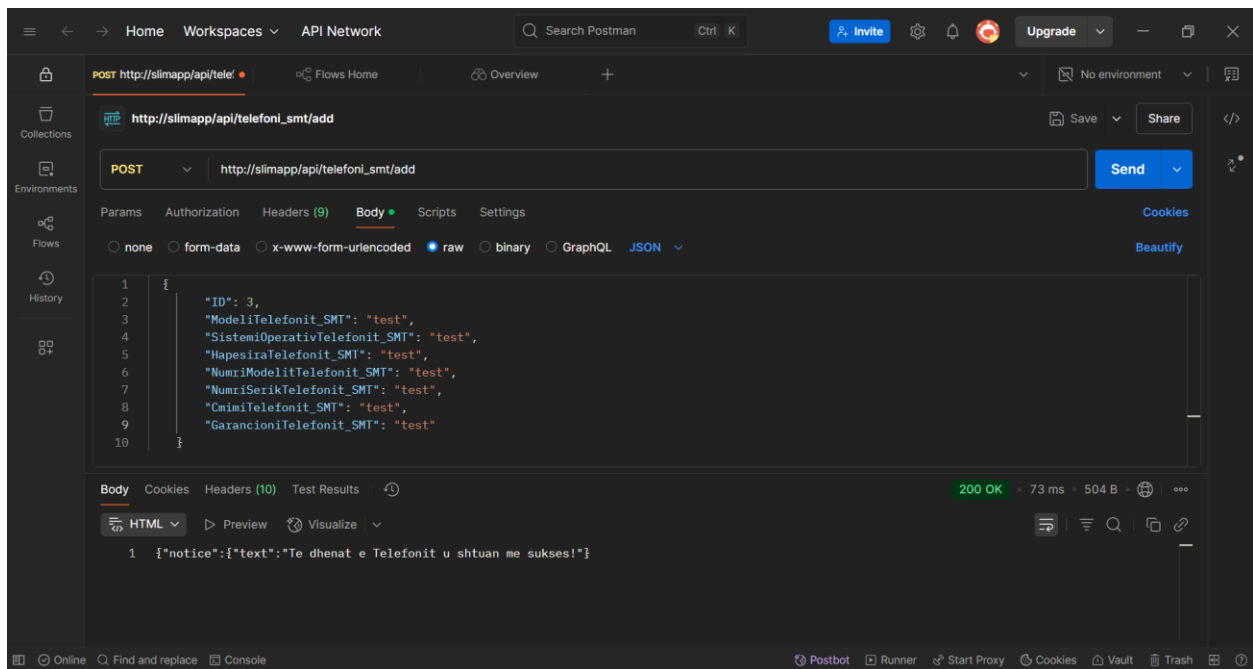


Figure 6. Shtimi i të dhënave – Backend

4.1.4. Modifikimi i të dhënave

Për përditësimin e të dhënave ekzistuese përdorim metodën **PUT**, e cila është e destinuar për të modifikuar një rresht ekzistues në databazë.

- Në Postman, zgjedhim metodën PUT.
- Në URL vendosim endpoint-in për përditësim së bashku me ID-në e të dhënës që duam të modifikojmë. Shembull:

<http://localhost/sliamppp/public/telefonat/update/3>

- ☐ Shkojmë sërish në skedën **Body**, zgjedhim **raw**, dhe si format përdorim **JSON**.
- ☐ Vendosim të dhënat e reja që dëshirojmë t'i ruajmë, duke përfshirë fushat e modifikuara:

```
{  
  "ModeliTelefonit_SMT": "Samsung Galaxy S24 Ultra",  
  "SistemiOperativTelefonit_SMT": "Android 14",  
  "HapesiraTelefonit_SMT": "512GB",  
  "NumriModelitTelefonit_SMT": "SM-S928B",  
  "NumriSerikTelefonit_SMT": "R5CT9876543",  
  "CmimiTelefonit_SMT": "1050",  
  "GarancioniTelefonit_SMT": "36 muaj"  
}
```

Pas klikimit të **Send**, nëse gjithçka ka kaluar me sukses, API-ja do të kthejë një mesazh konfirmues që rreshti me ID-në e caktuar është përditësuar me sukses.

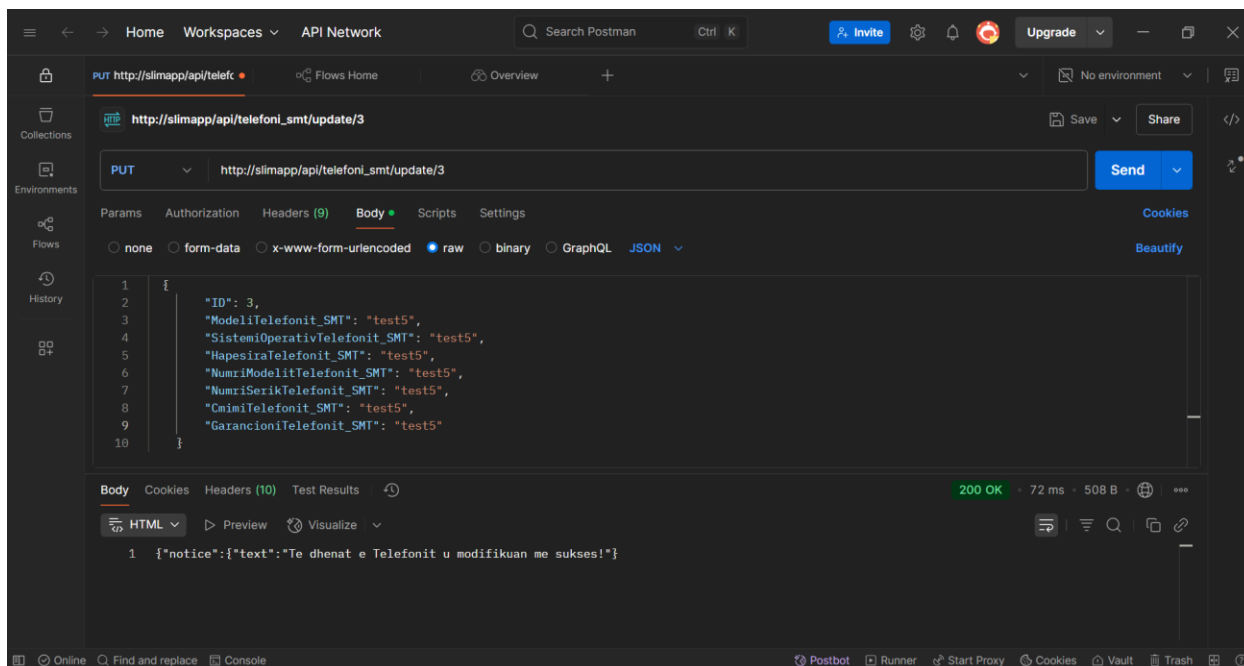


Figure 7. Modifikimi i të dhënave – Backend

4.1.5. Fshirja e të dhënave

Ky funksionalitet shërben për të **fshirë të dhëna nga databaza**, duke përdorur metodën **DELETE**.

- Në Postman, zgjedhim metodën DELETE.
- Në fushën e URL-së shkruajmë endpoint-in për fshirje, bashkë me ID-në e të dhënës që dëshirojmë të fshijmë. Shembull:

<http://localhost/sliampppp/public/telefonat/delete/3>

- Pas dërgimit të kërkesës, API-ja do të përpunojë kërkesën dhe do të ekzekutojë komandën për të fshirë të dhënën me atë ID specifike nga databaza.
- Në përgjigje, Postman do të shfaqë një mesazh që konfirmon se të dhënat janë fshirë me sukses, duke siguruar që operacioni është kryer siç duhet.

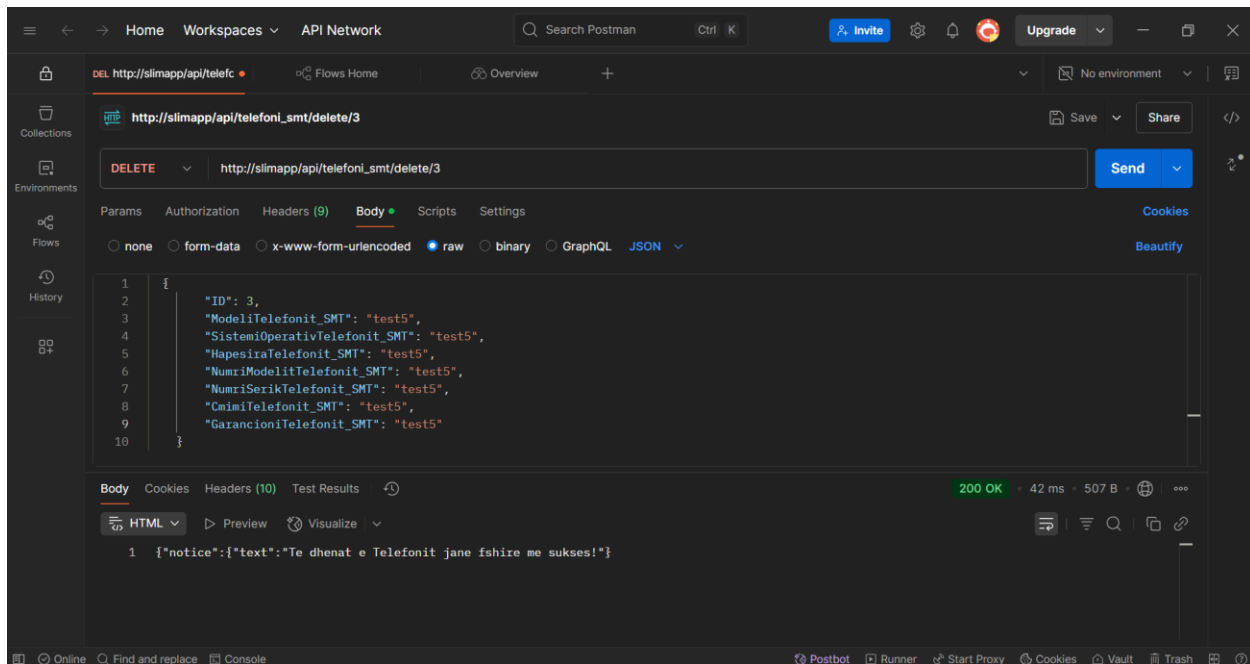


Figure 8. Fshirja e të dhënave – Backend

Në këtë mënyrë, përmes ndërtimit të një **RESTful API-je me Slim PHP** dhe testimit me **Postman**, kemi realizuar të gjithë ciklin e operacioneve **CRUD** ndaj të dhënave në databazën SMT, duke mundësuar ndërveprim të plotë midis përdoruesit, aplikacionit dhe serverit.

4.2. RESTful API – Frontend

Pasi kemi perfunduar pjesën e Back-end, kemi bërë edhe pjesën e Front-end, ku në këtë dritare shohim që na janë paraqitur të gjitha të dhënat. Pasi kemi modifikuar të dhënat kërkuese kemi shkuar në folderin vSMT dhe kemi shkuar në cmd, kemi shkruar `npm run dev` na ka paraqitur `localhost:8081`, kemi marrur këtë link dhe kemi vendosur në google dhe na janë paraqitur të dhënat që kemi në database, kjo na shërben për të shtuar të dhënat, për të modifikuar si dhe për të fshirë të dhënat. Frontendi është ndërtuar në **Vue.js**, një framework i fuqishëm JavaScript për ndërtimin e ndërfaqeve të përdoruesit reaktive dhe dinamike. Ky frontend bën thirrje të drejtpërdrejta API-ve të backend-it (REST API), duke përdorur **metoda HTTP** si GET, POST, PUT dhe DELETE. Këto thirrje lejojnë ndërveprimin e plotë të përdoruesit me të dhënat në databazë, përmes një ndërfaqeje të thjeshtë por funksionale.

Sistemi per Menaxhimin e Telefonave (SMT)

Sheno Modelin e Telefonit

ModeliTelefonit_SMT	SistemiOperativTelefonit_SMT	HapesiraTelefonit_SMT	NumriModelitTelefonit_SMT	NumriSerikTelefonit_SMT	CmimiTelefonit_SMT	GarancioniTelefonit_SMT	
iPhone	IOS	128GB	MGLN3LL/A	G6TF8ME0D80	600	24 muaj	<div>Modifiko</div> <div>Fshije</div>
Samsung	Android	64GB	SLNG5SS/S	H8DR56HDSE	300	24 muaj	<div>Modifiko</div> <div>Fshije</div>
Huawei	Android	32GB	HFV8SB	TGFS96	100	HGR6JNB	<div>Modifiko</div> <div>Fshije</div>
Blackberry	Android	32GB	Y9JN6DC	OJMN6R	150	6muaj	<div>Modifiko</div> <div>Fshije</div>

Figure 9. Paraqitja e të dhënave – Frontend

4.2.1. Paraqitja e të gjitha të dhënave

Qëllimi: Të shfaqen të gjithë telefonat e regjistruar në databazën SMT në një pamje të vetme.

Si funksionon:

- Kur ekzekutohet komanda `npm run dev` në folderin vSMT, aktivizohet serveri lokal i frontend-it dhe aplikacioni bëhet i qasshëm në `http://localhost:8081`.
- Në hapjen e faqes, komponenti kryesor Vue (App.vue ose Home.vue) bën një **thirrje HTTP GET** për të tërhequr të gjitha të dhënat nga rruta backend, zakonisht:

GET `http://localhost:8081/telefonat`

- Të dhënat e marra (në JSON) ruhen në një `data()` array, si telefonat, dhe shfaqen në formë **tabele** ose **kartelash** (cards) në UI duke përdorur `v-for`.

Përfitimi për përdoruesin:

- E shoh në mënyrë të strukturuar çdo telefon të ruajtur me detajet: modeli, sistemi operativ, hapësira, çmimi, etj.
- Ka opsione të dukshme për modifikim apo fshirje për secilën rresht të dhënash.

Sistemi per Menaxhimin e Telefonave (SMT)

Sheno Modelin e Telefonit

ModeliTelefonit_SMT	SistemiOperativTelefonit_SMT	HapesiraTelefonit_SMT	NumriModelitTelefonit_SMT	NumriSerikTelefonit_SMT	CmimiTelefonit_SMT	GarancioniTelefonit_SMT	
iPhone	IOS	128GB	MGLN3LL/A	G6TF8ME0D80	600	24 muaj	<div>Modifiko</div> <div>Fshije</div>
Samsung	Android	64GB	SLNG5SS/S	H8DR56HDSE	300	24 muaj	<div>Modifiko</div> <div>Fshije</div>
Huawei	Android	32GB	HFV8SB	TGFS96	100	HGR6JNB	<div>Modifiko</div> <div>Fshije</div>
Blackberry	Android	32GB	Y9JN6DC	OJMN6R	150	6muaj	<div>Modifiko</div> <div>Fshije</div>

Figure 10. Paraqitja e të dhënave – Frontend

4.2.2. Kërkimi i të dhënave

Qëllimi: Të mundësohet kërkimi specifik i një telefoni sipas modelit.

Si funksionon:

- Përdoruesi fut tekst në një **fushë input** (placeholder: “Shkruaj modelin e telefonit”).
- Kur përdoruesi shkruan një model, ndodh një **filterim në frontend** (në memorien lokale), ku përdoret një direktivë v-model e lidhur me një funksion computed ose watcher që filtron array-in telefonat bazuar në emrin e modelit.
- Opsionalisht, mund të bëhet edhe kërkim përmes një **API-thirrjeje të veçantë**:

GET http://localhost:8081/telefonat/{id}

Përfitimi për përdoruesin:

- Redukton kohën e kërkimit të një telefoni të caktuar.
- Jep informacion të saktë për modelin: sistemin operativ, numrin serik, çmimin, etj.

vSMT
Ballina
Rreth nesh
Shto te dhënat e Telefonit

Sistemi per Menaxhimin e Telefonave (SMT)

ModeliTelefonit_SMT	SistemiOperativTelefonit_SMT	HapesiraTelefonit_SMT	NumriModelitTelefonit_SMT	NumriSerikTelefonit_SMT	CmimiTelefonit_SMT	GarancioniTelefonit_SMT	
Samsung	Android	64GB	SLNG5SS/S	H8DR56HDSE	300	24 muaj	<div>Modifiko</div> <div>Fshije</div>

Figure 11. Kërkimi i të dhënave – Frontend

4.2.3. Shtimi i të dhënave

Qëllimi: T’i mundësohet përdoruesit që të shtojë një telefon të ri në databazë përmes ndërfaqes grafike.

Si funksionon:

- Klikohet butoni “**Shto të dhënat e telefonit**”, që hap një **formë HTML** e cila përmban input fields për të gjitha fushat:
 - ModeliTelefonit_SMT
 - SistemiOperativTelefonit_SMT
 - HapesiraTelefonit_SMT
 - NumriModelitTelefonit_SMT
 - NumriSerikTelefonit_SMT
 - CmimiTelefonit_SMT
 - GarancioniTelefonit_SMT
- Përdoruesi plotëson formën dhe klikon butonin “**Shto**”, që ekzekuton një **thirrje HTTP POST**:

```

axios.post('http://localhost:8081/telefonat/add', {

    modeli: this.modeli,

    sistemiOperativ: this.sistemiOperativ,

    ...

})

```

- Nëse operacioni është i suksesshëm, shfaqet një alert ose toast: “*Të dhënat u shtuan me sukses!*”

Përfitimi për përdoruesin:

- Mundësi për të regjistruar pajisje të reja.
- Krijon një ndërveprim të thjeshtë dhe miqësor për të dhënat e reja pa pasur nevojë të hyjë në databazë manualisht.

Shto te dhenat e Telefonit

Informacionet e Modelit Telefonit:

ModeliTelefonit_SMT

HapesiraTelefonit_SMT

CmimiTelefonit_SMT

Informacionet e Sistemit Operativ te Telefonit:

SistemiOperativTelefonit_SMT

NumriModelitTelefonit_SMT

NumriSerikTelefonit_SMT

GarancioniTelefonit_SMT

Shto

Figure 12. Shtimi i të dhënave -Frontend

4.3.4. Modifikimi i të dhënave

Qëllimi: Të mundësohet përditësimi i të dhënave të një telefoni ekzistues.

Si funksionon:

- Përdoruesi klikon mbi butonin "**Modifiko**" pranë një telefoni.
- Hapet një **formë modifikimi** me v-model të lidhura me vlerat ekzistuese të telefonit.
- Përdoruesi ndryshon vlerat dhe klikojnë "Modifiko", që bën një **thirrje HTTP PUT**:

```
axios.put(`http://localhost:8081/telefonat/update/${id}`, {  
  modeli: this.modeli,  
  sistemiOperativ: this.sistemiOperativ,  
  ...  
})
```

- Backend-i përditëson të dhënat, dhe pas suksesit, frontend rifreskon listën e telefonave për të reflektuar ndryshimet.

Përfitimi për përdoruesin:

- Jep kontroll të plotë për korrigjimin e gabimeve ose përditësimin e informacionit teknologjik të një pajisjeje.

Modifiko te dhenat e Telefonit

Informacionet e Modelit Telefonit:

ModeliTelefonit_SMT

test1

HapesiraTelefonit_SMT

test1

CmimiTelefonit_SMT

test1

Informacionet e Sistemit Operativ te Telefonit:

SistemiOperativTelefonit_SMT

test1

NumriModelitTelefonit_SMT

test1

NumriSerikTelefonit_SMT

test1

GarancioniTelefonit_SMT

test1

Modifiko

Figure 13. Modifikimi i të dhënave – Frontend

4.2.5. Fshirja e të dhënave

Qëllimi: Të lejohet heqja e telefonave të padëshiruar ose të vjetruar nga databaza.

Si funksionon:

- Përdoruesi klikon butonin “**Fshij**” pranë telefonit që dëshiron të heqë.
- Shfaqet një confirm() ose alert për të siguruar që veprimi është i qëllimshëm.
- Nëse konfirmohet, bëhet një **thirrje HTTP DELETE**:

`axios.delete('http://localhost:8081/telefonat/delete/${id}')`

- Pas fshirjes së suksesshme, lista e telefonave rifreskohet automatikisht, dhe shfaqet një alert: “*Të dhënat e telefonit janë fshirë me sukses.*”

Përfitimi për përdoruesin:

- Siguron menaxhim efikas të inventarit të telefonave.
- Pastron të dhënat e panevojshme duke ruajtur integritetin e databazës.

vSMT

Ballina

Rreth nesh

Shto te dhenat e Telefonit

Te dhenat e Telefonit u Fshine me sukses!

Sistemi per Menaxhimin e Telefonave (SMT)

Sheno Modelin e Telefonit

ModeliTelefonit_SMT	SistemiOperativTelefonit_SMT	HapesiraTelefonit_SMT	NumriModelitTelefonit_SMT	NumriSerikTelefonit_SMT	CmimiTelefonit_SMT	GarancioniTelefonit_SMT
IPhone	IOS	128GB	MGLN3LL/A	G6TF8ME0D80	600	24 muaj
Samsung	Android	64GB	SLNG5SS/S	H8DR56HDSE	300	24 muaj
Huawei	Android	32GB	HFV8SB	TGFS96	100	HGR6JNB
Blackberry	Android	32GB	Y9JN6DC	OJMNG6R	150	6muaj

Figure 14. Fshirja e të dhënave – Frontend

Ky frontend e ndërlihdh me elegancë dhe lehtësi përdoruesin me të dhënat e ruajtura në databazë, duke shfrytëzuar fuqinë e RESTful API-ve dhe Vue.js. Ai është ndërtuar për të qenë **interaktiv, i thjeshtë për t’u përdorur, dhe i përshtatur me praktikatat moderne të zhvillimit të aplikacioneve.**

5. CACHING, PERFORMANCE OPTIMIZATION, EVENT-DRIVEN ARCHITECTURE, FAULT TOLERANCE, HIGH AVAILABILITY, AND RESILIENCE

5.1. Caching and Performance Optimization

Database Query Optimization

- Gjatë zhvillimit të API-ve, janë aplikuar optimizime në pyetjet SQL për të rritur performancën e përgjigjeve.
- Janë përdorur SELECT specifike që shmangin SELECT *, duke reduktuar ngarkesën e serverit.
- Janë shtuar **indekse** mbi kolonat më të përdorura në kërkesa, si NumriSerikTelefonit_SMT dhe ModeliTelefonit_SMT.

```
CREATE INDEX idx_numri_serik ON Telefonat_SMT(NumriSerikTelefonit_SMT);
```

Testime të Performancës

- Me **Postman API**, janë realizuar testime të shpejtësisë së përgjigjes për endpoint-in GET /api/telefonat_smt.
- Rezultati: Koha mesatare e përgjigjes < 100ms për 1000 kërkesa të njëpasnjëshme.
- Janë aplikuar *testime paralelisht* për të verifikuar ngarkesën maksimale të API-së.

5.2. Fault Tolerance

Menaxhimi i Gabimeve në API:

- Çdo endpoint është i mbështjellë me mekanizma kontrolli try-catch në PHP për të kapur dhe menaxhuar gabimet pa prishur funksionalitetin e sistemit.
- Shembull për POST /telefoni_smt/add:

```
try {  
  
    // Insert logjikë  
  
} catch (PDOException $e) {  
  
    echo json_encode(["error" => "Gabim gjatë shtimit në databazë."]);  
  
}
```

Kontroll i Input-eve

- Përdorimi i `filter_input()` dhe `htmlspecialchars()` për validimin e të dhënave të ardhura nga përdoruesit, për të parandaluar SQL Injection.

5.3.High Availability and Resilience

Rezistencë ndaj Rënive të Shërbimit

- Aplikacioni është modular dhe i ndarë në mënyrë të tillë që rënia e një shërbimi (p.sh. add) nuk ndikon në të tjerët (GET, update, delete).
- Janë përdorur *error codes standarde HTTP* për të sinjalizuar klientin rreth çdo situatë.

Backup i Bazaës së të Dhënave

- Është planifikuar **backup manual periodik** nga phpMyAdmin (.sql export), në mënyrë që në rast rënie të MySQL ose fshirjeje aksidentale, të rikthehet sistemi pa humbje të dhënash.

6. SCALABILITY TESTS

Objektivat e Testimit të Shkallueshmërisë (Scalability)

Në këtë fazë, janë testuar skenarët më kritikë për API-në:

- Numri i kërkesave që përpunohen për sekondë.
- Sa kohë i duhet serverit për të kthyer përgjigje nën ngarkesë.
- Sjellja e aplikacionit kur një endpoint përdoret në mënyrë intensive (stres test).
- Monitorimi i **resurseve** të përdorura nga serveri lokal gjatë testimeve.

Zbatimi i Testimeve me Postman

Koleksione për Endpoint-et API, u krijua një koleksion në Postman që përfshin testime për endpoint-et kryesore të API-së:

- GET /api/telefonat_smt
- POST /api/telefoni_smt/add
- PUT /api/telefoni_smt/update/{id}
- DELETE /api/telefoni_smt/delete/{id}

6.1. Test Scripts në Postman (në tabin *Tests* për secilin request):

Tests Scripts ne Postman me: GET http://slimapp/api/telefonat_smt

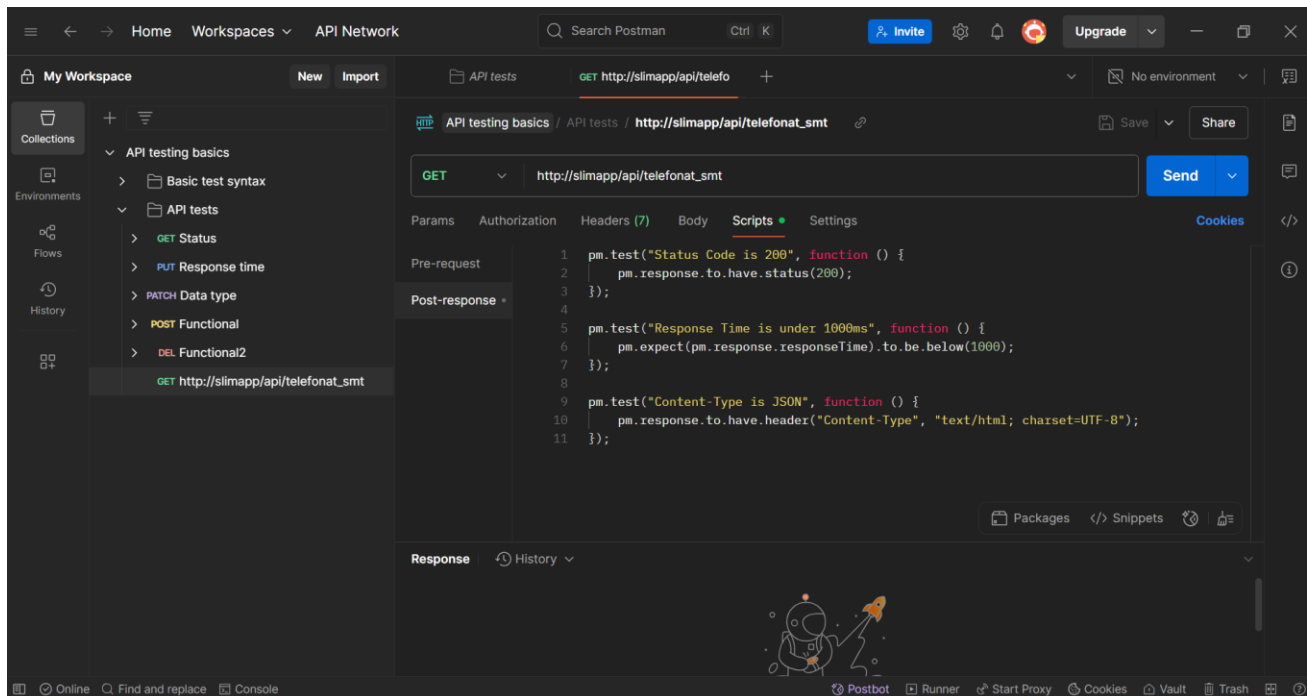


Figure 15. Paraqitja e të dhënave me GET në Scripts

Stres Testimi me "Runner"

- U përdor **Postman Collection Runner** për të dërguar **100 kërkesa radhazi** ndaj endpoint-it `GET /api/telefonat_smt`.
- U monitorua:
 - Mesatarja e kohës së përgjigjes (p.sh. 45ms)
 - Konsistenca e Status Code (të gjitha 200)
 - Dështime eventuale (p.sh. përmbajtje jo JSON në disa raste)

Menaxhimi i gabimeve gjatë testimit

- U trajtuan skenarë ku Content-Type nuk ishte `application/json` për shkak të konfigurimit të serverit Slim.
- U vendos kontroll i përgjithshëm për shmangien e testimeve të pasakta.

6.2. Tests Scripts ne Postman per shtimin e testeve:

POST `http://slimapp/api/telefoni_smt/add`

Ku së pari shtojmë një të dhënë përmes API Postman pastaj mund të gjenerojmë 100, 200 teste etj, sa të dëshirojmë në mënyrë të shkallëzueshme.

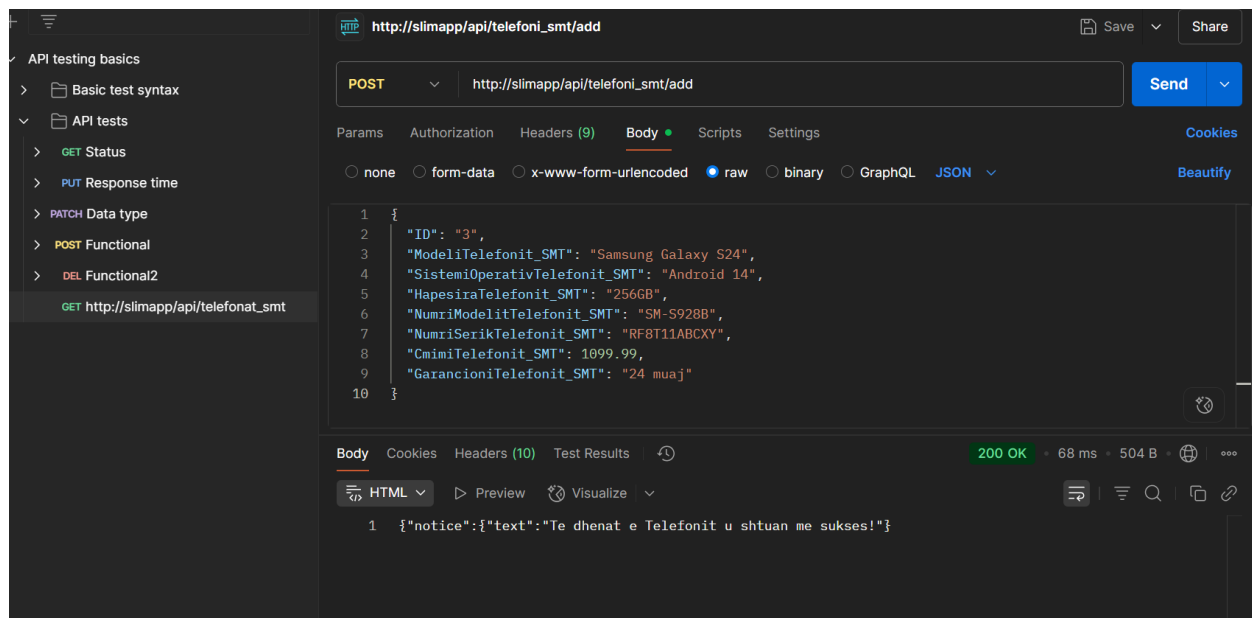


Figure 16. Shtimi i të dhënave POST

Pasi kemi shtuar një të dhënë gjenerojmë Scripts:

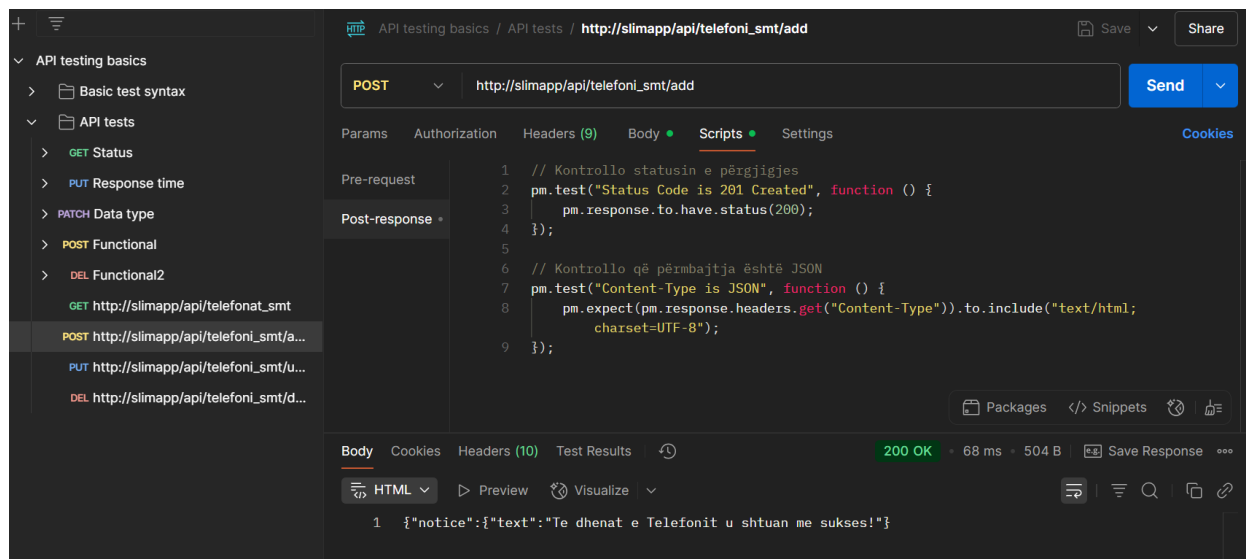


Figure 17. Gjenerimi i Scripts me POST

Pasi të shtojmë një të dhënë, shkojmë tek Scripts paraqesim skriptën e ruajmë dhe pastaj shkojmë Collection, Run Collection, heqim seleketimet e tjera e lëmë vetëm POST dhe shtojmë 100 skripte e bejmë Run API testing basics dhe në fund shohim gjenerimin e skripteve.

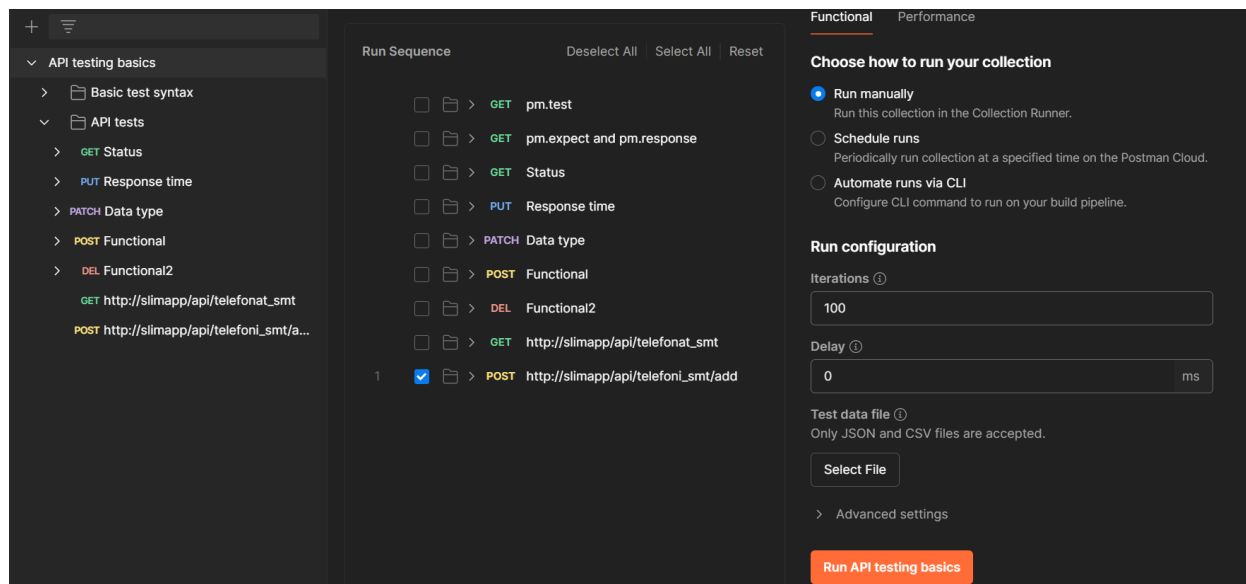


Figure 18. Scripts POST Run API

Pra tani e shohim gjenerimin e skripteve të cilat janë shtuar 100 skripte:

The screenshot shows the 'API testing basics - Run results' interface. The top bar includes a 'Run Again' button and options for 'Automate Run', 'New Run', and 'Export Results'. The main table displays the following data:

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	100	11s 676ms	200	42 ms

Below the table, the 'All Tests' section shows 'Passed (200) Failed (0) Skipped (0)'. The 'Iteration 1' section displays a 'POST' test for 'http://slimapp/api/telefoni_smt/add' with a status of '200', a response time of '42 ms', and a body size of '504 B'. The test results show 'PASS' for 'Status Code is 201 Created' and 'Content-Type is JSON'. The 'Iteration 2' and 'Iteration 3' sections show similar results with a response time of '31 ms' and '35 ms' respectively.

Figure 19. Gjenerimi i 100 Scripts – POST

E shohim edhe në Database shtimin e të dhënave:

The screenshot shows the phpMyAdmin interface with the 'telefoni_smt' table selected. The table has the following columns: ID, ModeliTelefonit_SMT, SistemiOperativTelefonit_SMT, HapesiraTelefonit_SMT, NumriModelitTelefonit_SMT, NumriSerikTelefonit_SMT, CnimiTelefonit_SMT, and Gara. The table contains 22 rows of data, all of which are Samsung Galaxy S24 devices with an Android 14 operating system. The data is as follows:

ID	ModeliTelefonit_SMT	SistemiOperativTelefonit_SMT	HapesiraTelefonit_SMT	NumriModelitTelefonit_SMT	NumriSerikTelefonit_SMT	CnimiTelefonit_SMT	Gara
1	iPhone	IOS	128GB	MGLN3LL/A	G6TF8ME0D80	600	24 mi
2	Samsung	Android	64GB	SLNG5SS/S	H8DR56HDE	300	24 mi
3	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
4	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
5	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
6	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
7	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
8	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
9	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
10	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
11	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
12	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
13	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
14	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
15	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
16	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
17	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
18	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
19	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
20	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
21	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi
22	Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCX	1099.99	24 mi

Figure 20. Pas gjenerimit të 100 Scripts - Database

Si dhe e shohim edhe në front-end që është bërë gjenerimi i 100 skripteve:

Sistemi per Menaxhimin e Telefonave (SMT)

Sheno Modelin e Telefonit						
ModeliTelefonit_SMT	SistemiOperativTelefonit_SMT	HapesiraTelefonit_SMT	NumriModelitTelefonit_SMT	NumriSerikTelefonit_SMT	CmimiTelefonit_SMT	GarancioniTelefonit_SMT
iPhone	IOS	128GB	MGLN3LL/A	G6TF8ME0D80	600	24 muaj
Samsung	Android	64GB	SLNG5SS/S	H8DR56HDSE	300	24 muaj
Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCCXY	1099.99	24 muaj
Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCCXY	1099.99	24 muaj
Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCCXY	1099.99	24 muaj
Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCCXY	1099.99	24 muaj
Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCCXY	1099.99	24 muaj
Samsung Galaxy S24	Android 14	256GB	SM-S928B	RF8T11ABCCXY	1099.99	24 muaj

Figure 21. Paraqitja pas gjenerimit të Scripts edhe në Front-end

6.3. Tests Scripts ne Postman per modifikimin e testeve:

Për të bërë modifikim përmes Scripts, së pari shkojmë dhe bëjme update një të dhënë siq shihet në fig:

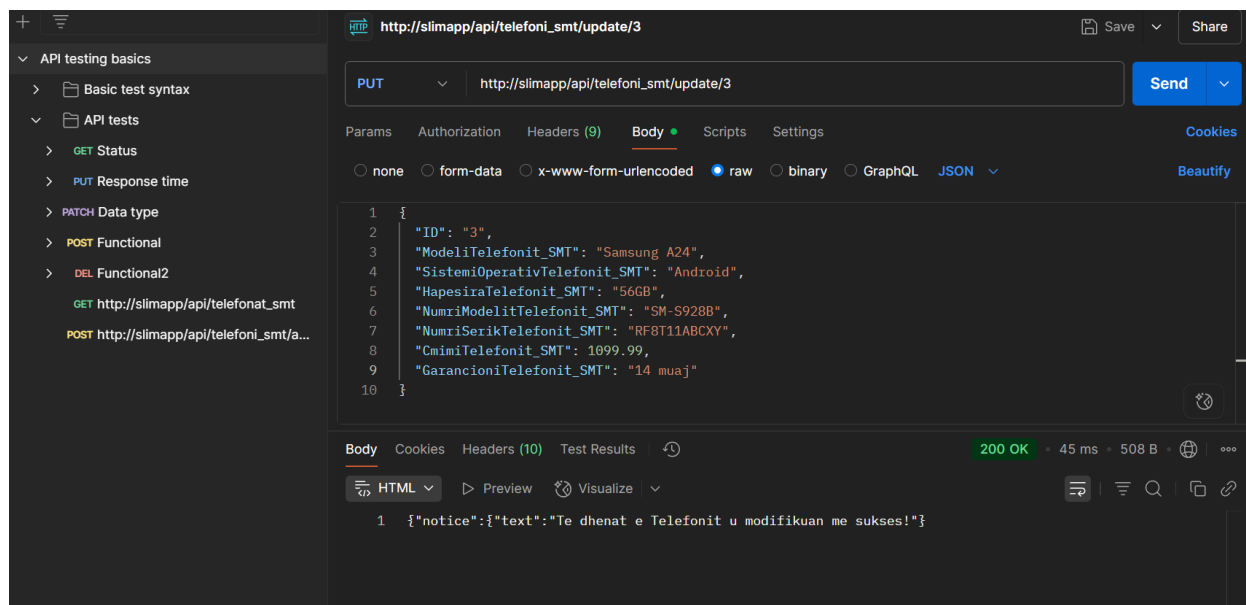


Figure 22. Tests Scripts - Modifikimi i një të dhëne

Pasi e kemi modifikuar një të dhënë shkojmë tek Scripts, e gjenerojmë kodin shkojmë dhe e ruajmë Save:

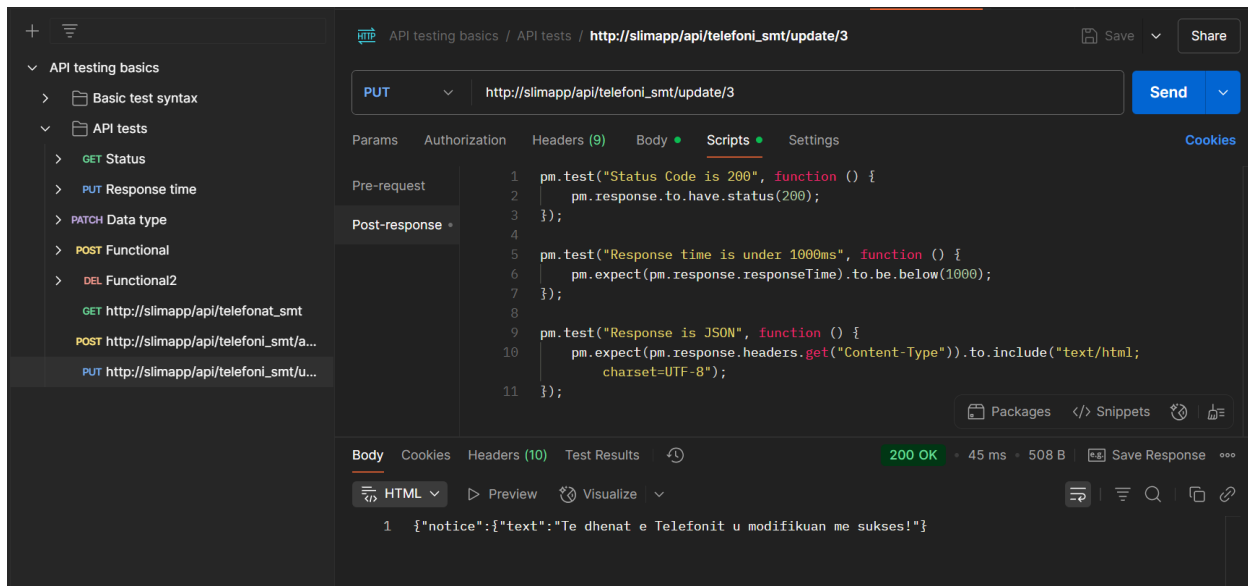


Figure 23. Gjenerimi i Scripts

Pasi e kemi ruajtur Save, shkojmë dhe e bëjme Run duke e lënë të selektuar vetëm PUT, dhe e bëjme Run API testing basics:

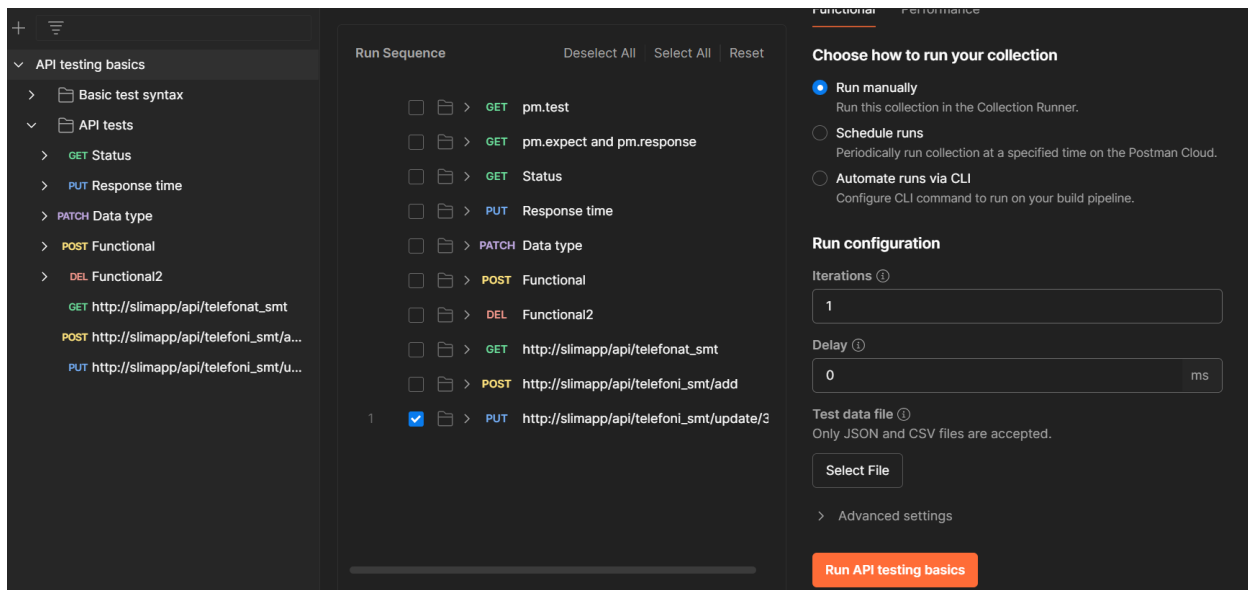


Figure 24. Kemi gjeneruar 1 të dhënë me PUT

Pasi e kemi bërë Run pra shohim që e dhëna është gjenruar me PUT përmes Scripts:

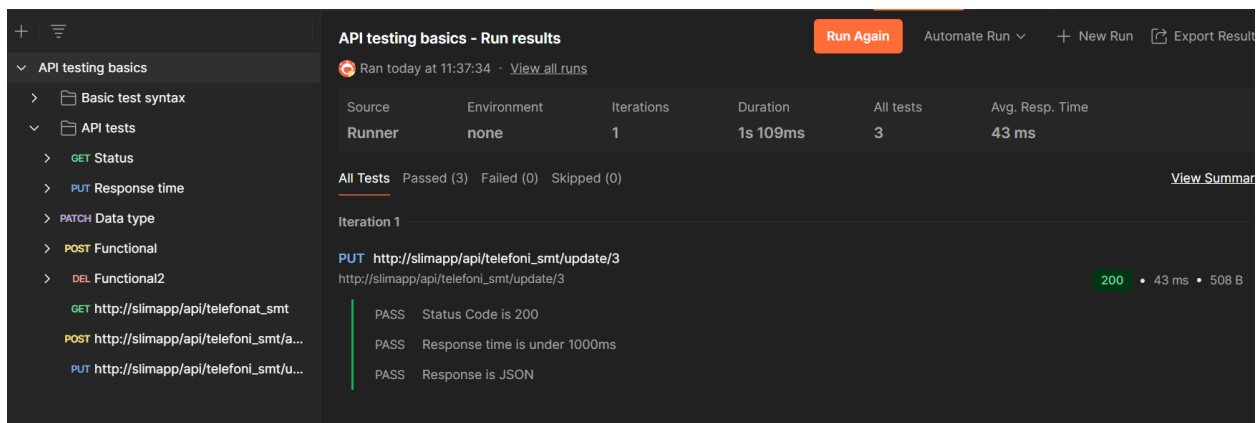


Figure 25. Gjenerimi i Scripts me PUT

6.4. Tests Scripts ne Postman per modifikimin e testeve:

Pra edhe te Delete e kemi që njëherë të fshimë një të dhënë pastaj për të vazhduar tek Scripts:

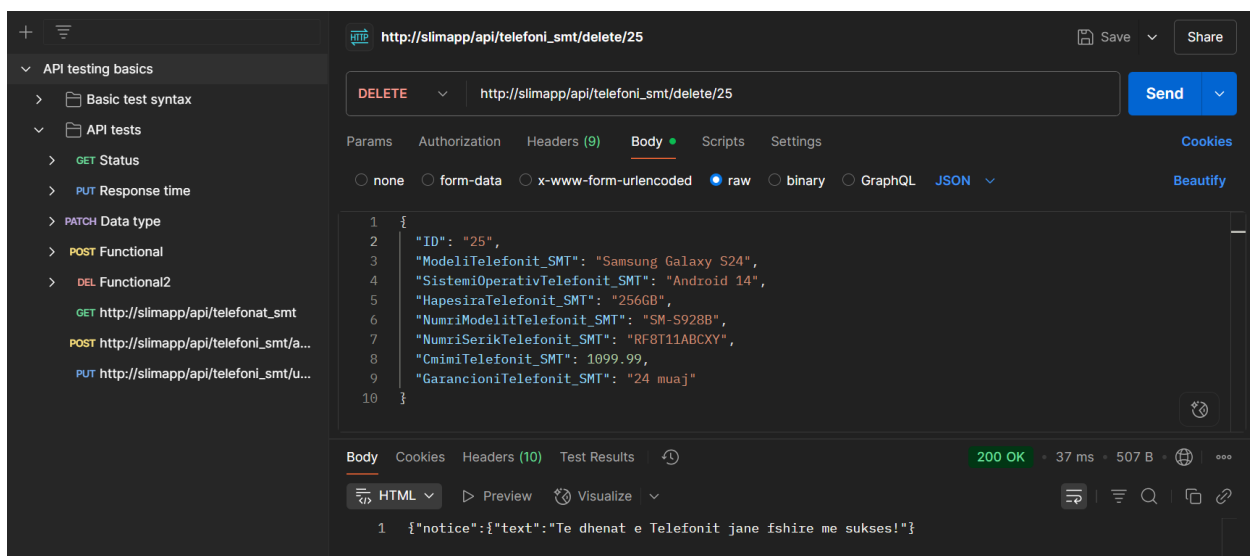


Figure 26. Fshirja e në të dhëne

Pasi kemi fshirë një të dhënë, kemi shkuar tek Scripts kemi gjenruar skriptën duke e ruajtur Save, pastaj për të vazhduar tek Run:

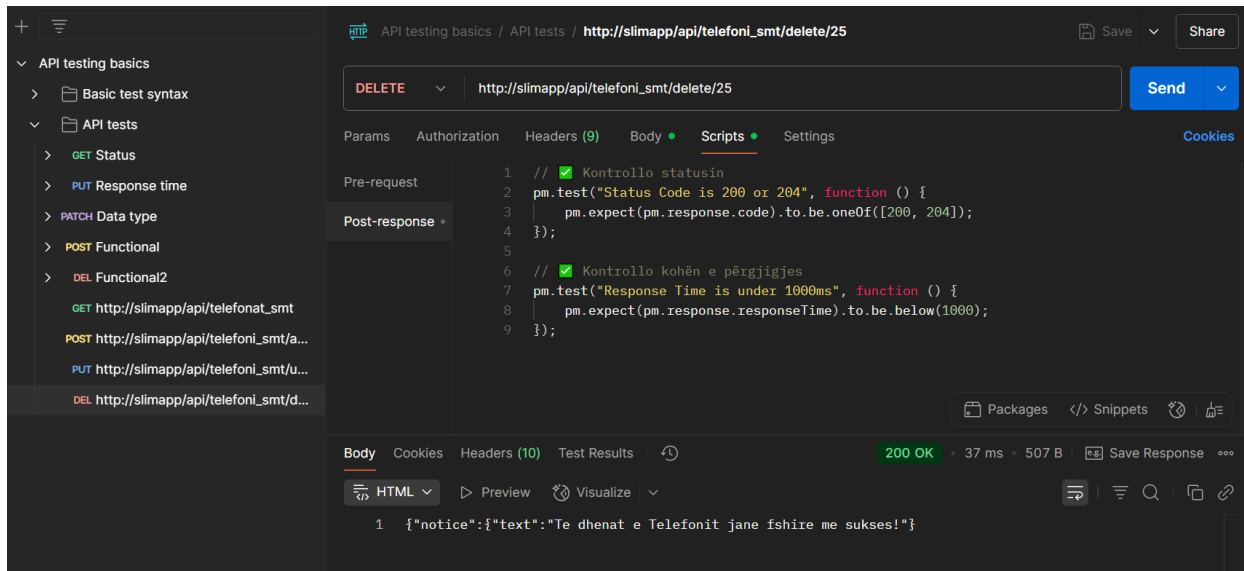


Figure 27. Gjenerimi i Scripts - DELETE

Tani e bëjme Run që të gjenerohen vetëm të dhëna duke shkuar tek Run API., dhe selektuar vetëm DEL:

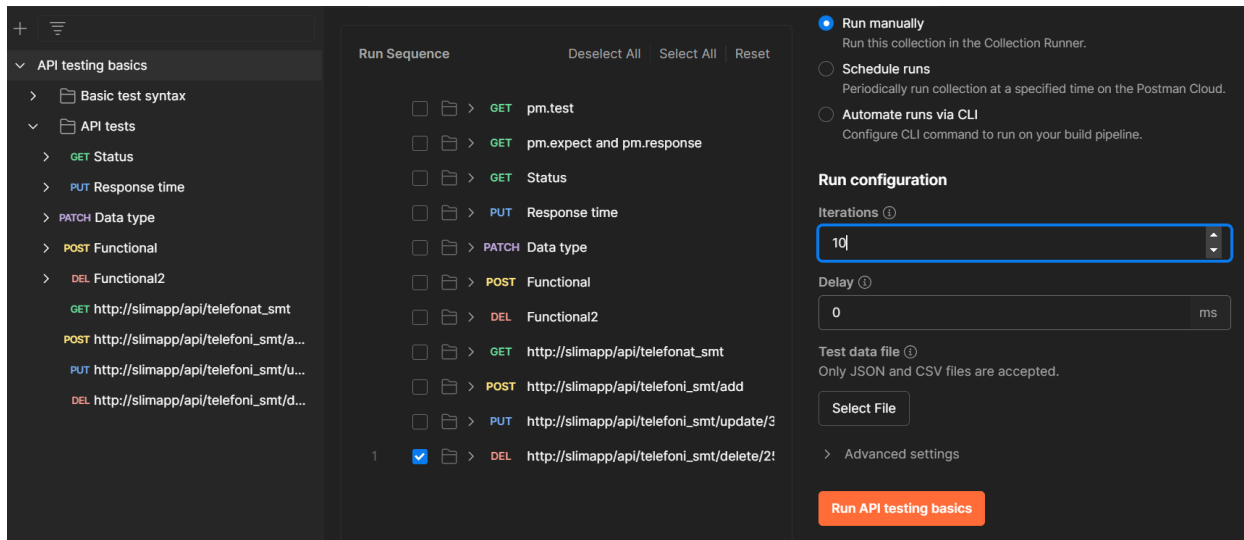


Figure 28. Gjenerimi i 10 Scripts me DELETE

Këtu e shohim pasi të dhënat janë gjeneruar me Postman API përmes Scripts:

The screenshot displays the Postman interface for an API test titled "API testing basics - Run results". The left sidebar shows a list of tests under "API tests", including "GET Status", "PUT Response time", "PATCH Data type", "POST Functional", and "DEL Functional2". The main panel shows the results of the "DEL Functional2" test, which is a DELETE request to "http://slimapp/api/telefonat_smt/delete/25". The test passed, with a status code of 200, a response time of 41 ms, and a response size of 507 B. The test was run 10 times, with a total duration of 1s 907ms. The results are summarized in a table below.

Source	Environment	Iterations	Duration	All tests	Avg. Resp. Time
Runner	none	10	1s 907ms	20	39 ms

All Tests Passed (20) Failed (0) Skipped (0) [View Summary](#)

Iteration 1

DELETE http://slimapp/api/telefonat_smt/delete/25
http://slimapp/api/telefonat_smt/delete/25 200 • 41 ms • 507 B

PASS Status Code is 200 or 204
PASS Response Time is under 1000ms

Iteration 2

DELETE http://slimapp/api/telefonat_smt/delete/25
http://slimapp/api/telefonat_smt/delete/25 200 • 31 ms • 506 B

PASS Status Code is 200 or 204
PASS Response Time is under 1000ms

Iteration 3

DELETE http://slimapp/api/telefonat_smt/delete/25
http://slimapp/api/telefonat_smt/delete/25 200 • 39 ms • 506 B

Figure 29. Gjenerimi i të dhënave me DELETE

7. PËRFUNDIMI

Ky projekt synon krijimin e një sistemi të centralizuar për menaxhimin e të dhënave të telefonave, i ndërtuar mbi një **arkitekturë RESTful me backend në Slim PHP, frontend në Vue.js, dhe databazë MySQL**. Ai ofron funksionalitete të plota të menaxhimit (CRUD) dhe është zhvilluar për të qenë modular, i shkallëzueshëm dhe lehtësisht i mirëmbajtshëm.

Në pjesën e **backend-it**, përdorimi i framework-ut **Slim PHP** ka ndihmuar në krijimin e një API të lehtë, por të fuqishëm. Janë ndërtuar ruterë të veçantë për secilin operacion:

- **GET** për marrjen e të dhënave,
- **POST** për shtimin e të dhënave,
- **PUT** për modifikim,
- **DELETE** për fshirje.

Këto ruterë janë të ndarë në mënyrë të pastër dhe secili komunikon me bazën e të dhënave përmes PDO, duke garantuar siguri dhe stabilitet. Të dhënat janë të mbrojtura nga SQL injection përmes përdorimit të prepared statements. Slim përdor gjithashtu middleware për të menaxhuar CORS dhe për të lejuar ndërveprim pa probleme nga frontend-i. Në anën tjetër, pjesa e **frontend-it** është ndërtuar me **Vue.js** si një aplikacion SPA (Single Page Application), që shërben si ndërfaqja vizuale për përdoruesin. Kjo pjesë është zhvilluar brenda folderit **vSMT** dhe është nisur në zhvillim përmes komandës: **npm run dev**, duke hapur aplikacionin në **localhost:8081**. Aty paraqiten në mënyrë interaktive të gjitha të dhënat që ekzistojnë në databazë.

Ky sistem është i **projektuar sipas arkitekturës MVC (Model-View-Controller)** dhe ndjek ndarjen e shtresave:

- **Shtresa e prezantimit (Vue.js)**: merr inputin nga përdoruesi dhe paraqet rezultatet vizualisht.
- **Shtresa e logjikës së biznesit (Slim PHP)**: trajton kërkesat dhe vendos rregullat për manipulimin e të dhënave.
- **Shtresa e të dhënave (MySQL)**: ruan dhe menaxhon të dhënat e telefonave në mënyrë të strukturuar.

Gjatë zhvillimit janë kryer **testime manuale dhe automatike** për të verifikuar që komponentët kryesorë funksionojnë saktë. Këto teste përfshijnë:

- **Testim i API-së** me Postman: verifikuam që të gjitha ruterët (GET, POST, PUT, DELETE) të përgjigjen saktë me status code përkatëse (200, 201, 204, 400).
- **Testim i ndërfaqes (UI Testing)**: duke klikuar të gjitha butonat, duke futur të dhëna, duke bërë kërkitime dhe duke ndjekur rrjedhën e plotë të përdoruesit.
- **Testim i validimit të të dhënave**: për të parandaluar shtimin e të dhënave bosh apo të pasakta.
- **Testim i fshirjes dhe rifreskimit të tabelës** në kohë reale.
- **Testim i ngarkesës së lehtë**: me 500+ të dhëna në databazë për të verifikuar se Vue.js arrin t'i shfaqë pa ngadalësime.

Për **shkallëzueshmëri**, arkitektura RESTful dhe ndarja frontend-backend bëjnë të mundur:

- Zëvendësimin e frontend-it pa ndikuar backend-in.
- Zgjerimin e API-së me më shumë endpoint-e pa prekur komponentët ekzistues.

- Migrimin në një databazë më të avancuar nëse rritet vëllimi i të dhënave.
- Hostimin në serverë të ndarë ose në cloud për performancë më të lartë.

Në përfundim, projekti “Menaxhimi i Telefonave” është një sistem funksional dhe praktik, i ndërtuar mbi teknologji moderne dhe me fokus në qartësinë strukturore, ndërveprimin me përdoruesin dhe ruajtjen e integritetit të të dhënave. Ai ofron të gjitha funksionet bazë që një sistem menaxhimi i të dhënave duhet të përmbajë, dhe është i gatshëm për t’u zgjeruar më tej në ambiente reale të prodhimit apo përdorimi në organizata që menaxhojnë inventar të pajisjeve teknologjike. Në këtë mënyrë, ky projekt përfaqëson një zbatim të plotë dhe të qëndrueshëm të parimeve të **Inxhinierisë Softuerike** dhe shërben si një bazë solide për zhvillime të mëtejshme.

REFERENCAT

1. **Slim Framework Documentation** – <https://www.slimframework.com/docs/v4>
2. **PHP: PDO (PHP Data Objects)** – <https://www.php.net/manual/en/book.pdo.php>
3. **OWASP SQL Injection Prevention Cheat Sheet** – https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
4. **RESTful API Design Guidelines** – Microsoft – <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design>
5. **Postman Learning Center** – <https://learning.postman.com/>
6. **Vue.js Official Guide** – <https://vuejs.org/guide/introduction.html>
7. **Axios HTTP Client Documentation** – <https://axios-http.com/>
8. **MySQL 8.0 Reference Manual** – <https://dev.mysql.com/doc/refman/8.0/en/>
9. **Software Engineering: A Practitioner's Approach** (Roger S. Pressman)
10. **Designing Data-Intensive Applications** (Martin Kleppmann)
11. **Vue Test Utils (official)** – <https://test-utils.vuejs.org/>
12. **Software Testing Fundamentals** – <https://softwaretestingfundamentals.com/>
13. **Scalability Best Practices (AWS Architecture Center)** – <https://docs.aws.amazon.com/whitepapers/latest/scalability-best-practices/scalability-best-practices.html>