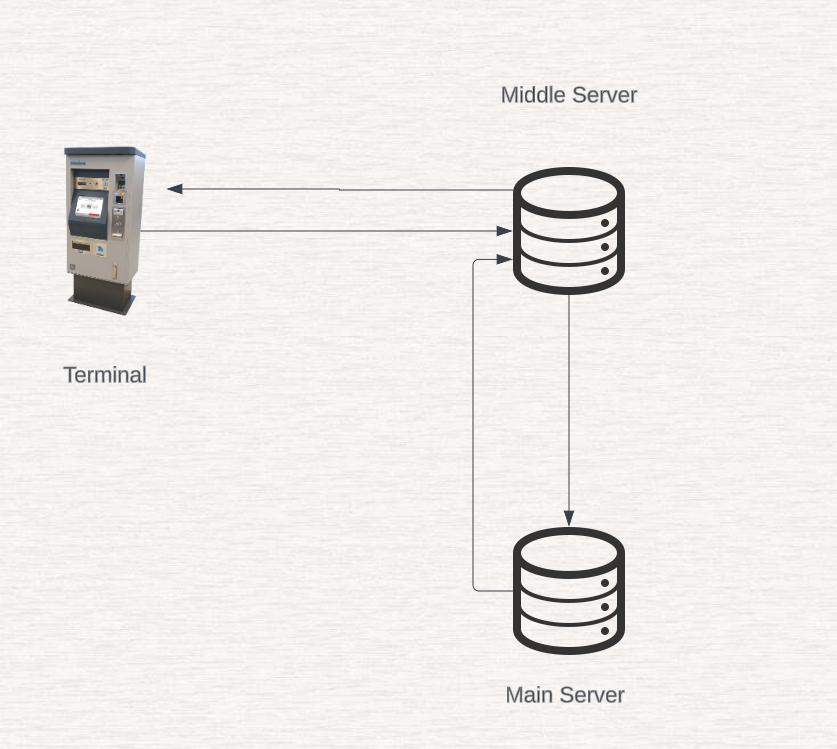
**Բիզնես պահանջ**

Անհրաժեշտ է նկարագրել API, որին կինտեգրվեն վճարային տերմինալները վարկի մարումներն իրականացնելու համար:

Կատարող։ Դավիթ Մուրադյան

API-ի ընդհանուր սխեման նկարագրված է վար նշված պատկերում որը տեխնիկական տեսանկյունից ավելի մանրամասն կբացատրվի կոդի օրինակով։ Հարկ է նշել որ API-ը կատարում է կապող օղակի դերը ծրագրերի մրջև։



Իրական պատկերը իր մեջ պարունակում է բազմաթիվ կոմպոնենտներ որոնք նշված չեն տվյալ սխեմայում, սակայն ընդհանուր գաղափարը հասկանալու համար տվյալ օրինակը շատ լավ ներկայացնում է գլխավոր կետերը։

Ինչպես տեսնում ենք վճարային տերմինալի և գլխավոր տվյալների բազայի միջև կա միջնորդ բազա որը կատարում է շատ կարևոր դեր։ Այն օգնում է նվազեցնել ծանրաբեռնվածությունը հիմնական տվյալների բազայի վրա և կատարում է քեշինգ-ի ֆունկցիան տվյալներին ավելի արագ հասանելիություն ստանալու նպատակով։

Մինչ անցնելը հիմնական քայլերին, հարկավոր է նշել որ օրինակի նպատակով գրված API-ի համար օգտագործվել է **Flask** 7 **requests** գրադարանները։

Քայլերի հերթականությանը հետևելով կկարողանանք ավելի մանրամասն պատկերացում կազմել API-ի աշխատանքի մասին։

Վար նշված աղյուսակում պատկերված է հրամանների մի մասը թե ինչպես են API-ները աշխատում HTTP հարցումներով։

|  |  |
| --- | --- |
| POST | Ստեղծել |
| GET | Ստանալ |
| PUT | Ամբողջական թարմացում |
| PATCH | Նշված կետերի թարմացում |
| DELETE | Ջնջել |

Կոդում oգտագործվող HTTP կարգավիճակներ

|  |  |
| --- | --- |
| 200 | Հաջողված հարցում |
| 400 | Թերի հարցում (պակաս ինֆորմացիա) |
| 404 | Գոյություն չունի |

1. Հաճախորդը տերմինալից օգտվելուց մուտքագրում է վարկի id-ն վարկի նույնականացման համար (այդ id-ն եզակի է և թույլ է տալիս տվյալների բազայում գտնել նշված վարկի տվյալները)
2. Տերմինալը այդ id-ն ուղարկում է middle server HTTP GET հարցում /get\_loan\_info վերջնակետին
3. Middle Server-ը ստուգում է արդյոք իր մեջ ունի տվյալ վարկի ինֆորմացիան թե ոչ։ Ինֆորմացիայի առկայության դեպքում այն միանգամից ուղարկում է ինֆորմացիան դեպի տերմինալ, ցուցադրելով այն
4. Ինֆորմացիայի բացակայության դեպքում middle server-ը GET հարցում է անում Main Server-ից (տվյալները փոխանցվում են JSON ֆորմատով)
5. Main Server-ը ստանալով հարցումը իր մեջ փնտրում է ինֆորմացիան և առկայության դեպքում փոխանցում middle server-ին
6. Middle Server- ը ստանալով ինֆորմացիան քեշավորում է իր մեջ հետագա հարցումների համար
7. Վերջապես ունենալով ամբողջ ինֆորմացիան այն ցուցադրվում է Տերմինալի էկրանին որտեղ երևում է այն ինֆորմացիան որը մենք ցանկանում ենք ցուցադրել հաճախորդին։ Օրինակ ՝ անուն, ազգանուն, վարկի մարման գումարը տվյալ ժամանակաշրջանի համար, մարման ենթակա ամբողջ գումարը և այլն

MAIN\_SERVER\_URL = 'http://localhost:5000'

@app.route('/get\_loan\_info', *methods*=['GET'])

def get\_loan\_info():

    loan\_id = request.args.get('loan\_id')

    if not loan\_id:

        return jsonify({"error": "Loan ID is required"}), 400

    # ստուգել ինֆորմացիայի առկայությունը

    if loan\_id in loan\_cache:

        return jsonify(loan\_cache[loan\_id]), 200

    # եռե առկա չէ ինֆորմացիան ապա ստանալ այն հիմնական սերվերից

    response = requests.get(f'{MAIN\_SERVER\_URL}/get\_loan\_info', *params*={'loan\_id': loan\_id})

    if response.status\_code == 200:

        # քեշավորում է վարկի տվյալները

        loan\_cache[loan\_id] = response.json()

    return jsonify(response.json()), response.status\_code

1. Ըստ ստացված ինֆորմացիայի, հաճախորդը կատարում է վճարումը և տերմինալը ուղարկում է HTTP POST հարցում Middle Server-ին /pay\_loan վերջնակետին
2. Middle Server-ը վճարման հարցումը ուղարկում է Main Server-ին /pay\_loan վերջնակետին
3. Main Server-ը թարմացնում է վարկի ինֆորմացիան և ուղարկում նոր տվյալները դեպի middle server որտեղ կատարվում է քեշավորում
4. Այդ ամենից հետո middle Server-ը ուղարկում է ինֆորմացիան դեպի տերմինալ գործարքի հաջողությամբ կատարման մասին (կամ խնդրի եթե առկա է)

Ամբողջական API կոդը Main և Middle Server-ի ինչպես նաև հավելյալ ֆայլերը կարող եք տեսնել GitHub-ի միջավայրում հետևյալ հղումով։ <https://github.com/MuDavid/API.git>