O‘ZBEKISTON RESPUBLIKASI OLIY TA‘LIM, FAN VA INNOVATSIYALAR VAZIRLIGI

FARG‘ONA DAVLAT TEXNIKA UNIVERSITETI

“Axborot texnologiyalari va telekommunikatsiya” fakulteti

“Dasturiy injiniring” kafedrasi

“BackEnd ilovalar yaratish” fanidan

MUSTAQIL ISH

Bajardi: **Abdumutalipov Sh**

650-22-guruh

Qabul qildi: **Xayitov A**

Farg‘ona 2025

1-mavzu: Backend tushunchasi – server va mijoz o‘rtasidagi aloqa, oddiy “Hello World” server yaratish

REJA

1. Backend dasturlash tushunchasi va uning zamonaviy axborot tizimidagi o‘rni

2. Server va mijoz (Client–Server Architecture) o‘rtasidagi aloqa tamoyillari

3. HTTP protokoli: asosiy tushunchalar, metodlar va ishlash mexanizmi

4. Backend texnologiyalari: Node.js, Python (Django/Flask), Java (Spring), PHP (Laravel) haqida umumiy ma’lumot

5. Amaliy qism: oddiy “Hello World” server yaratish (Node.js va Python misolida)

1. Backend dasturlash tushunchasi va uning zamonaviy axborot tizimidagi o‘rni

Zamonaviy raqamli ekotizimda dasturiy ta’minot ikki asosiy qatlamga bo‘linadi: foydalanuvchi bilan bevosita muloqot qiluvchi **frontend** qatlami va ma’lumotlar, biznes mantiq hamda serverdagi jarayonlarni boshqaruvchi **backend** qatlami. Backend dasturlash — bu foydalanuvchining ko‘ziga ko‘rinmaydigan, ammo butun tizimning barqaror ishlashini ta’minlab turadigan asosiy vosita bo‘lib, serverda bajariladigan hisob-kitoblar, ma’lumotlar bazasi bilan ishlash, autentifikatsiya, avtorizatsiya, xavfsizlik, API yaratish kabi jarayonlarni o‘z ichiga oladi.

Backend tushunchasi dastlab 1960–1970-yillarda yaratilgan dasturiy tizimlarda paydo bo‘lgan bo‘lib, u davrda serverlar markaziy komputerlar sifatida ishlagan. Bugungi kunda esa backend internet xizmatlarining yuragi hisoblanadi. Amazon, Google, Netflix, Meta kabi gigant kompaniyalarning asosiy qiymati aynan ularning backend infratuzilmasi samaradorligiga bog‘liq. Turli manbalarda (masalan, **Martin Kleppmann – “Designing Data-Intensive Applications”**, **RFC 7231 – HTTP Specification**) backend serverlar yuqori darajadagi ishonchlilik, uzluksizlik va masshtablanuvchanlik tamoyillariga asoslanishi aytiladi.

Backend tizimlarning o‘rni haqida gap ketganda, zamonaviy web va mobil ilovalarning deyarli barchasi majburiy ravishda backend xizmatiga tayanishini

ta’kidlash lozim. Masalan, foydalanuvchi Telegram yoki Instagram kabi ilovalardan biror kontentni yo‘qotmasdan foydalanishi uchun barcha ma’lumotlar markaziy serverlarda saqlanadi. Backend server foydalanuvchi yuborgan so‘rovlarni qabul qiladi, kerakli ma’lumotlarni tekshiradi, ma’lumotlar bazasiga murojaat qiladi va natijani qaytaradi. Shu jarayonda backendning ishonchli ishlashi butun xizmatning sifatiga bevosita ta’sir qiladi.

Backend strukturasida asosiy komponentlar mavjud:

1. **Server** — backend kodlari bajariladigan muhit. Bu fizik kompyuter, virtual server yoki bulutli xizmat bo‘lishi mumkin.
2. **Ma’lumotlar bazasi** — barcha axborotlar saqlanadigan joy. MySQL, PostgreSQL, MongoDB kabi texnologiyalar keng qo‘llanadi.
3. **API (Application Programming Interface)** — frontend yoki boshqa ilovalarga xizmat ko‘rsatish interfeysi.
4. **Biznes mantiq** — foydalanuvchi so‘rovlariga ishlov berishdagi qoidalar, algoritmlar.
5. **Xavfsizlik moduli** — autentifikatsiya, avtorizatsiya, shifrlash, xavfsizlik devorlariga oid mexanizmlar.

Backend tizimlarining bugungi dasturchilik infratuzilmasidagi o‘rni ayniqsa mikroxizmatlar (microservices) arxitekturasining ommalashuvi bilan yanada mustahkamlandi. Netflix, Uber, Amazon Web Services singari xizmatlar butun tizimni yuzlab, ba’zan minglab kichik servislar bo‘linmali tarzda quradi. Har bir servis alohida backend sifatida ishlaydi va API orqali o‘zaro muloqot qiladi. Bu esa tizimni masshtablashni osonlashtiradi, xizmatlarning uzluksizligini oshiradi.

Shuningdek, backend dasturlash sun’iy intellekt, katta ma’lumotlar (Big Data) va bulutli texnologiyalar (AWS, Google Cloud, Azure) bilan chambarchas bog‘liq. Katta hajmdagi ma’lumotlar oqimini qayta ishlash, real vaqt rejimida xizmat ko‘rsatish, millionlab foydalanuvchilarga bir vaqtning o‘zida xizmat berish backend serverlarning imkoniyatiga bog‘liq.

Yana bir muhim jihat — backendning xavfsizlikdagi roli. Dunyo miqyosida Google Developers, OWASP (Open Web Application Security Project) kabi tashkilotlar backend xavfsizligini ta’minlash bo‘yicha standart va tavsiyalar ishlab chiqadi. Masalan, OWASP Top 10 ro‘yxatida backend orqali sodir bo‘ladigan eng keng tarqalgan xatoliklar (SQL Injection, XSS, Authentication flaws) keltiriladi. Shuning uchun backend dasturchi har doim xavfsizlik standartlariga rioya qilishi lozim.

Bugungi IT mehnat bozorida backend dasturchilarga talab yuqori. LinkedIn, Indeed, Glassdoor kabi platformalardagi statistikaga ko‘ra, backend bo‘yicha mutaxassislar 2023–2024 yillarda eng talabgir yo‘nalishlardan biri bo‘lgan. Chunki har qanday raqamli mahsulot — internet do‘kon, banking ilovasi, ta’lim platformasi, tibbiyot tizimi — backendga tayanadi.

Xulosa qilib aytganda, backend — bu dasturiy ta’minotning ko‘zga ko‘rinmaydigan, ammo funksionallikni to‘liq ta’minlab turuvchi asosiy qatlami bo‘lib, zamonaviy axborot tizimlarining boshqaruvi, ma’lumotlar almashinuvi va ish jarayonining uzluksizligini kafolatlaydi.

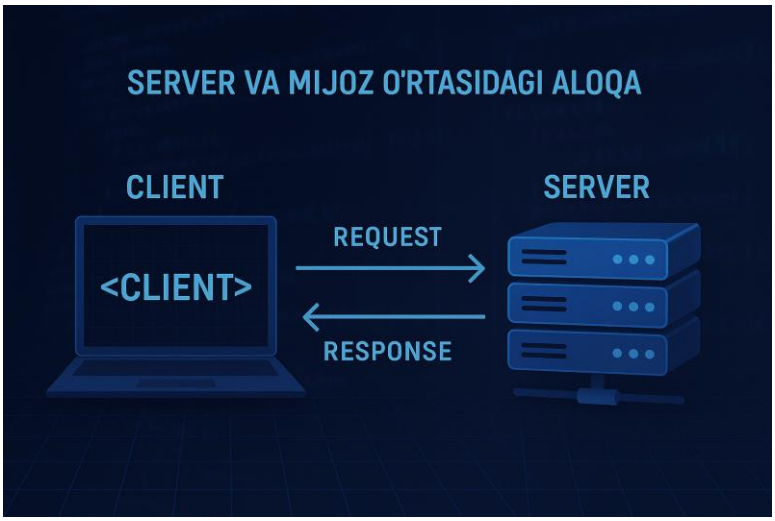
2. Server va mijoz (Client–Server Architecture) o‘rtasidagi aloqa tamoyillari — Yangi, uzluksiz matnli variant

Zamonaviy axborot texnologiyalari rivojida “client–server” arxitekturasi eng muhim tushunchalardan biri bo‘lib, u internet orqali ma’lumot almashinuvi jarayonining bosh mexanizmini tashkil etadi. Bu modelning paydo bo‘lishi 1980–1990-yillarga borib taqaladi va u dastlab korporativ tarmoqlarda qo‘llanilgan bo‘lsa-da, bugungi kunda barcha web-servislar, mobil ilovalar, bulutli xizmatlar va hatto IoT qurilmalari aynan shu tamoyil asosida ishlamoqda. Client–server modeli mohiyatan shuni bildiradiki, tarmoqda ikki xil rolga ega bo‘lgan ikki ishtirokchi mavjud bo‘ladi: biri — mijoz, ya’ni foydalanuvchi nomidan so‘rov yuboruvchi dastur; ikkinchisi esa — server, ya’ni kelgan so‘rovlarni qayta ishlab, tegishli javobni qaytaruvchi markaziy hisoblash tuguni. Andrew Tanenbaumning “Computer Networks” asarida aytilishicha,

ushbu arxitektura tizimlarning funksiyalarini ajratish orqali ularni boshqarishni yengillashtiradi va ko‘p foydalanuvchili tizimlarda barqarorlikni oshiradi.

Mijozning vazifasi oddiy ko‘rinishi mumkin, ammo u juda muhim bosqichlardan iborat. Mijoz foydalanuvchi tomonidan berilgan buyruqlarni dastlab o‘z tizimida qayta ishlaydi, masalan, web brauzer manzil satriga kiritilgan URL manzilini DNS orqali tekshiradi, ushbu domenning haqiqiy IP manzilini topadi va shundan so‘ng serverga so‘rov yuboradi. Mijoz odatda yengil dastur hisoblanadi, chunki u ma’lumotlar ustida murakkab amallar bajarmaydi; uning vazifasi kerakli manzilga to‘g‘ri formatlangan HTTP so‘rovi yuborish va serverdan kelgan natijani foydalanuvchiga qulay ko‘rinishda namoyish qilishdan iborat. Eng keng tarqalgan mijoz turiga web brauzerlar misol bo‘ladi; Chrome, Firefox, Safari kabi brauzerlar aynan shu mexanizm orqali ishlaydi.

Server esa butun tizimning asosiy hisoblash markazi bo‘lib, mijozlardan kelgan so‘rovlarga aniq javob qaytarish uchun ishlab chiqilgan. Server doimiy ravishda faol holatda turadi va har bir so‘rovni ketma-ket yoki parallel tarzda qayta ishlaydi.



Serverlarda ma’lumotlar qayta ishlanadi, bazadan kerakli ma’lumotlar olinadi, autentifikatsiya va xavfsizlik jarayonlari amalga oshiriladi. Serverlar uzluksiz ishlashi uchun ular kuchli apparat resurslari bilan ta’minlanadi va ko‘pincha ularning bir nechta nusxalari yaratiladi, bu esa yukni teng taqsimlash imkonini beradi. Shu boisdan serverlar ko‘pincha yirik data-markazlarda joylashtiriladi va ular maxsus monitoring, xavfsizlik va avtomatlashtirilgan boshqaruv tizimlari yordamida nazorat qilinadi.

Client–server o‘zaro aloqasi odatda HTTP yoki HTTPS protokollari orqali amalga oshadi. Bu protokollarning ishlash tamoyili RFC hujjatlarida (masalan, RFC

7230 va RFC 2616) batafsil bayon qilingan. HTTP protokoli mijoz tomonidan yuborilgan matnli so‘rovni serverga yetkazadi va server qaytargan matnli javobni mijozga uzatadi. Bu jarayonda URL manzili orqali resurs aniqlanadi, so‘rovning turi belgilanadi va server tegishli javobni qaytaradi. Misol tariqasida, brauzer biror rasm, matn yoki faylni so‘raganda server uni topib, HTTP javobi ko‘rinishida qaytaradi va brauzer uni foydalanuvchi uchun vizual ko‘rinishga keltiradi. Bu jarayon bir necha millisekund davomida sodir bo‘ladi. Internet sahifalarining tez yuklanishi ham aynan serverning qay darajada optimallashtirilganiga bog‘liq.

Client–server arxitekturasi ko‘p qatlamli modelga ega bo‘lib, bunda mijoz faqat ma’lumotni ko‘rish va boshqarish bilan shug‘ullanadi, ilova qatlamida esa biznes jarayonlari amalga oshiriladi. Eng past qatlamda ma’lumotlar bazasi mavjud bo‘lib, server ushbu baza bilan uzluksiz muloqotda bo‘ladi. Ushbu model Microsoft, IBM, Google kabi kompaniyalar tomonidan ishlab chiqilgan arxitektura qo‘llanmalarida eng samarali tizimlardan biri sifatida e’tirof etilgan. Chunki u tizimning tashkiliy murakkabligini kamaytiradi, modullarning mustaqilligini ta’minlaydi va xavfsizlikni yuqori darajaga ko‘taradi.

Client–server modeli ayniqsa internet xizmatlari kengayishi bilan sezilarli ustunliklarni namoyon qildi. Uzluksiz ishlash, ma’lumotlarning markazlashgan holda boshqarilishi, xavfsizlikning yaxshilanishi, xizmatlarning masshtablanishi va mijozlar uchun qulaylik yaratilishi uni amalda eng asosiy arxitekturalardan biriga aylantirdi. Bugungi kunda ijtimoiy tarmoqlar, davlat portallari, online banking tizimlari, ta’lim platformalari, o‘yin serverlari, hatto global tarmoqlarning o‘zagi bo‘lgan DNS tizimlari ham aynan mijoz va server o‘rtasidagi aniq belgilangan kommunikatsiya tamoyillariga asoslanadi.

Xulosa sifatida aytish mumkinki, client–server arxitekturasi — zamonaviy raqamli dunyoning bosh mexanizmidir. Har bir xizmat, har bir web sahifa, har bir mobil ilova foydalanuvchi bilan server o‘rtasida uzluksiz axborot almashinuvi tufayli ishlaydi. Bu arxitektura nafaqat internetning bugungi ko‘rinishini, balki uning

kelajakdagi rivojlanish yo‘nalishini ham belgilab beruvchi fundamental texnologik asos hisoblanadi.

3. HTTP protokoli: asosiy tushunchalar, metodlar va ishlash mexanizmi

Zamonaviy internet xizmatlarining deyarli barchasi o‘z faoliyatini HTTP protokoliga tayangan holda amalga oshiradi. HyperText Transfer Protocol — qisqacha HTTP deb ataluvchi ushbu protokol 1991-yilda Tim Berners-Li tomonidan World Wide Web loyihasi uchun yaratilgan bo‘lib, bugungi kunda global tarmoqning asosiy kommunikatsiya qatlamlaridan biriga aylangan. HTTP protokoli mohiyatan mijoz va server o‘rtasida matnli so‘rovlar almashinuvi orqali resurslarni uzatish mexanizmi bo‘lib, u internetda mavjud bo‘lgan barcha web sahifalar, API xizmatlar, mobil ilovalar va hatto IoT qurilmalarning ma’lumot uzatish jarayonlarida qo‘llaniladi. Protokolning ishlash tartibi va texnik talablari eng avvalo IETF tomonidan qabul qilingan RFC 2616, RFC 7230, RFC 7231 kabi hujjatlarda batafsil yoritilgan bo‘lib, ular protokolning qanday ishlashi, qanday xabar formatlari qo‘llanishi va qanday javob kodlari mavjudligini aniq belgilab beradi.

HTTP protokoli oddiy ko‘rinishga ega bo‘lsa-da, uning zamirida mijoz va server o‘rtasidagi muloqotni standartlashtiruvchi kuchli mexanizm yotadi. Mijoz tomonidan yuborilgan HTTP so‘rovi matndan iborat bo‘lib, unda URL manzil, so‘rov turi, qo‘shimcha sarlavhalar (headers) hamda ba’zan qo‘shimcha ma’lumotlar (body) bo‘lishi mumkin. Server ushbu so‘rovni qabul qiladi, uni tahlil qiladi va tegishli javobni shakllantiradi. HTTP javobi ham matnli bo‘lib, unda status kodi, sarlavhalar va resursning tarkibi mavjud bo‘ladi. Mijoz va server o‘rtasidagi ushbu almashinuv TCP/IP tarmoq arxitekturasi asosida amalga oshiriladi, ya’ni HTTP o‘zi mustaqil tarmoq protokoli bo‘lmay, yuqori darajadagi protokol sifatida TCP ustida ishlaydi.

HTTP protokolining eng muhim xususiyatlaridan biri — uning stateless tabiati hisoblanadi. Bu esa har bir so‘rov mustaqil ekanini anglatadi: server mijozning avvalgi so‘rovlaridagi holatni eslab qolmaydi. Bunda tizimning soddaligi va moslashuvchanligi ta’minlansa-da, autentifikatsiya kabi jarayonlarda qo‘shimcha mexanizmlarni qo‘llash

zarur bo‘ladi. Shu maqsadda cookie, session yoki tokenlar (masalan, JWT) ishlatiladi. Stateless tamoyili HTTP protokolining yengil, tez va resurslarni kam talab qiluvchi tizim sifatida ommalashishiga sabab bo‘lgan.

HTTP protokoli o‘zining rivojlanish bosqichlarida bir necha versiyalarni boshdan kechirdi. Dastlabki HTTP/0.9 juda sodda bo‘lib, faqat bitta tipdagi so‘rovni qo‘llagan. keyinchalik HTTP/1.0 va ayniqsa HTTP/1.1 versiyasi internet muhitida asosiy standartga aylangan. Bugungi kunda esa HTTP/2 va HTTP/3 kabi zamonaviy protokollar ishlab chiqildi. HTTP/2 binary formatdan foydalanishi, bir nechta so‘rovlarni parallel ravishda bir kanal orqali uzata olishi tufayli web sahifalarning sezilarli darajada tez yuklanishini ta’minlaydi. HTTP/3 esa QUIC protokoli orqali UDP asosida ishlaydi, bu esa yuqori kechikishli tarmoqlarda ham barqarorlikni oshiradi. Barcha ushbu versiyalar internetdagi katta xizmatlar — Google, YouTube, Amazon, Cloudflare kabi platformalarda amalda qo‘llanilib kelmoqda.

HTTP protokolida ishlatiladigan metodlar internetdagi muloqotning qanday turda amalga oshirilishini aniqlaydi. Eng ko‘p qo‘llaniladigan GET metodi resursni olish uchun xizmat qiladi. Mijoz brauzer orqali sahifani ochganda aynan GET so‘rovi yuboriladi. POST esa serverga ma’lumot yuborish, masalan, ro‘yxatdan o‘tish yoki forma yuborish jarayonlarida qo‘llaniladi. PUT resursni yangilash, DELETE esa uni o‘chirish uchun mo‘ljallangan. REST arxitektura tamoyillarida aynan shu metodlar ma’lumotlar bilan ishlashning tartibli usulini belgilaydi va bu API xizmatlarini yaratishda standartlashgan yondashuvni shakllantiradi. Eng muhimi, barcha metodlar semantik ma’noga ega bo‘lib, ular serverda bajariladigan harakatlarni aniq tasvirlaydi.

HTTP protokolining ishlash mexanizmi bir qarashda murakkab ko‘rinsa-da,aslida u aniq tartibga ega mantiqiy jarayondan iborat. Mijoz avval TCP ulanishi o‘rnatadi, undan so‘ng to‘liq shakllangan HTTP so‘rovini yuboradi. Server ushbu so‘rovni bosh qismidan boshlab o‘qiydi, kerakli resursni aniqlaydi va talab etilgan amalni bajaradi. Agar resurs topilmagan bo‘lsa, server javob sifatida 404 status kodini qaytaradi. Agar ichki xatolik yuz bersa, 500 kodi yuboriladi. Agar hammasi yaxshi

bo‘lsa, 200 yoki shu ma’noga yaqin bo‘lgan status kodi qaytadi. Status kodlari protokolning eng muhim elementlaridan biri bo‘lib, ular mijozga jarayonning qaysi bosqichida qanday holat yuz berganini aniq ko‘rsatadi. Shu boisdan internetdagi barcha brauzerlar va API vositalari status kodlarini to‘liq qo‘llab-quvvatlaydi.

Tushuncha	Mazmuni (kitob uslubida ilmiy ta’rif)
HTTP Protokoli	HyperText Transfer Protocol — internetda mijoz va server o‘rtasida matnli xabarlar orqali ma’lumot almashinuvi uchun mo‘ljallangan yuqori darajadagi protokol bo‘lib, uning texnik standartlari RFC 2616 va RFC 7230 seriyali hujjatlar bilan belgilangan.
Stateless Prinsipi	HTTP so‘rovlari bir-biridan mustaqil bo‘lib, server avvalgi so‘rovlar tarixini eslab qolmaydi. Bu tamoyil tarmoq resurslaridan yengil foydalanishni ta’minlaydi, ammo autentifikatsiya jarayonlarida qo‘shimcha mexanizmlarni talab qiladi.
Request (So‘rov)	Mijoz tomonidan serverga yuboriladigan matnli xabar bo‘lib, unda metod, manzil, sarlavhalar va ba’zan ma’lumot qismi mavjud. So‘rov serverga ma’lum bir amalni bajarish uchun ko‘rsatma beradi.
Response (Javob)	Server tomonidan mijozga yuboriladigan xabar bo‘lib, sonli status kodi, sarlavhalar va qaytarilayotgan resurs tarkibidan iborat. Javob serverda so‘rov qanday qayta ishlanganini aks ettiradi.
HTTP Metodlari	Protokol doirasida bajariladigan semantik amallarni belgilovchi buyruqlar bo‘lib, GET resursni olishni, POST yangi ma’lumot yuborishni, PUT o‘zgartirishni, DELETE o‘chirishni bildiradi.
Status Kodlari	Server javobining holatini bildiruvchi uch xonali sonlar tizimi bo‘lib, 200–299 oralig‘i muvaffaqiyatli, 400–499 mijoz xatosi, 500–599 esa server xatosini ifodalaydi.
Headers (Sarlavhalar)	Mijoz va server o‘rtasida texnik va qo‘shimcha ma’lumotlarni uzatuvchi elementlar bo‘lib, ular orqali ma’lumotning turi, uzunligi, formatlash usuli, autentifikatsiya ma’lumotlari va boshqa parametrlar uzatiladi.
Body (Ma’lumot qismi)	So‘rov yoki javobning mazmuniy tarkibi bo‘lib, unda JSON, XML, HTML yoki boshqa formatdagi ma’lumotlar bo‘lishi mumkin. U asosan POST, PUT kabi metodlarda faol qo‘llaniladi.
HTTP/1.1	Web muhitda eng keng qo‘llanilgan versiya bo‘lib, persistent connections, chunked transfer encoding kabi imkoniyatlarni taqdim etadi.
HTTP/2	Binary formatga asoslangan, ko‘p oqimli uzatishni qo‘llovchi, sahifa yuklanish tezligini sezilarli oshiruvchi zamonaviy versiya bo‘lib, Google va Amazon servislarida faol qo‘llaniladi.
HTTP/3 (QUIC)	UDP asosida ishlaydigan, yuqori kechikishli tarmoqlar uchun optimallashtirilgan protokol bo‘lib, global darajadagi xizmatlarda ishlatiladi.

HTTP sarlavhalari (headers) protokolning alohida o‘ta muhim qismi bo‘lib, ular orqali mijoz va server o‘rtasida qo‘shimcha texnik ma’lumotlar yuboriladi. Masalan,

User-Agent sarlavhasi mijozning qurilmasi va brauzeri haqida ma’lumot beradi; Content-Type sarlavhasi yuborilayotgan ma’lumotning turini bildiradi; Authorization sarlavhasi esa himoyalangan resurslarga kirish uchun zarur bo‘lgan autentifikatsiya ma’lumotlarini uzatadi. Bularning barchasi HTTP orqali ma’lumot almashinuvi jarayonida aniqlik, tartib va xavfsizlikni ta’minlaydi. Ayniqsa, zamonaviy API xizmatlarining ko‘pchiligi JSON formatidan foydalanadi va server “application/json” tipidagi sarlavhani qaytarib, mijozga ma’lumotning qanday formatda ekanini bildiradi.

Muxtasar qilib aytganda, HTTP protokoli internetning qon tomirlari vazifasini bajaradi. U holda internet xizmatlari, web sahifalar, mobil ilovalar, hatto tarmoqdagi qurilmalar o‘rtasidagi muloqot ham imkonsiz bo‘lar edi. Protokolning ochiq standartlarga asoslangani, yengil va universalligi, turli qurilmalar va operatsion tizimlar bilan to‘liq moslashuvchanligi uni global miqyosdagi eng asosiy kommunikatsiya mexanizmlaridan biriga aylantirgan. Shu sababli backend dasturchi HTTP protokoli, uning metodlari, status kodlari, sarlavhalari va versiyalarini chuqur bilishi zarur. Bu bilimlar web ilova yaratish jarayonida ham, API dizayn qilishda ham, server arxitekturasini ishlab chiqishda ham asosiy mezon sifatida xizmat qiladi.

4. Backend texnologiyalari: Node.js, Python (Django/Flask), Java (Spring), PHP (Laravel) haqida umumiy ilmiy-nazariy ma’lumot

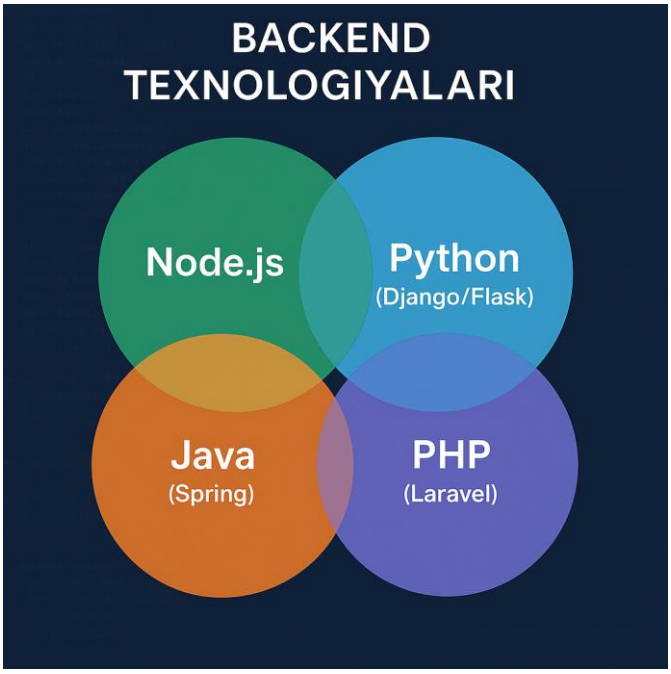
Zamonaviy dasturiy ta’minot arxitekturasida backend texnologiyalari fundamental o‘rin egallaydi. Backend — bu ilovaning ko‘zga ko‘rinmaydigan, ammo barcha ma’lumotlar, hisoblashlar va xizmat mantiqlari bajariladigan vazirlik darajasidagi asosiy qatlamdir. Backend texnologiyalari vaqt o‘tishi bilan sezilarli evolyutsiyani boshdan kechirdi. Dastlab server tomonida faqat oddiy skript tili bo‘lgan PHP kabi vositalar ishlatilgan bo‘lsa, bugungi kunda yuqori yuklamalarga bardosh bera oladigan, mikroxizmatlarga bo‘linadigan, bulutli muhitlarga moslashgan murakkab platformalar paydo bo‘lgan. Shu sababli backend texnologiyalarini nafaqat sintaksis darajasida, balki arxitektura, ishlash mexanizmi va ularning rivojlanish falsafasi nuqtai nazaridan ham chuqur anglash muhimdir.

Node.js — bu backend texnologiyalari rivojida tub o‘zgarish qilgan vosita bo‘lib, 2009-yilda Ryan Dahl tomonidan yaratilgan. Uning asosiy innovatsiyasi JavaScript tilini server tomonida ishlatish imkoniyatini yaratganida emas, balki uning “non-blocking I/O” tamoyiliga asoslanganida edi. Node.js voqealarga asoslangan arxitekturaga ega bo‘lib, bitta ipda ishlasa-da, minglab bir vaqtning o‘zida kelayotgan so‘rovlarni samarali qayta ishlay oladi. Bu xususiyat uni real vaqt rejimidagi xizmatlar — chatlar, onlayn o‘yinlar, monitoring tizimlari, WebSocket bilan ishlaydigan ilovalar uchun juda mos qiladi. Node.js V8 JavaScript engine asosida ishlaydi, bu esa uning ishlash tezligini yuqori darajaga ko‘taradi. Yana bir muhim jihati — npm ekotizimining boyligi bo‘lib, u millionlab tayyor paketlarni taklif qiladi. Bu imkoniyat backend dasturchilarga murakkab funksiyalarni qisqa vaqt ichida qo‘shish imkonini beradi.

Python backendda qo‘llaniladigan eng mashhur texnologiyalardan biri bo‘lib, uning asosiy kuchi sodda sintaksis, barqaror kutubxonalar va kuchli hamjamiyatdadir. Pythonning backend bo‘yicha ikki asosiy freymvorki — Django va Flask hisoblanadi. Django — “batteries included” falsafasiga asoslanib yaratilgan yirik tizim bo‘lib, unda autentifikatsiya, admin panel, router tizimi, ORM, xavfsizlik mexanizmlari, caching kabi komponentlar allaqachon tayyor holatda mavjud. Bu yirik kompaniyalar — Instagram, Pinterest, Spotify kabi xizmatlarda aynan Django qo‘llanilishiga sabab bo‘lgan. Flask esa minimalistik yondashuvga asoslangan bo‘lib, u dasturchiga juda katta erkinlik beradi. Flask’ning yengilligi uni mikroxizmatlarni yaratish, kichik API lar, ilmiy yoki eksperimental loyihalarda qo‘llash uchun mos qiladi. Python’ning backenddagi afzalligi shundaki, uning ekotizimi sun’iy intellekt, mashinaviy o‘qitish, ma’lumotlar tahlili bilan yaxshi integratsiyalashgan bo‘lib, bu zamonaviy server funksiyalarini yanada kuchaytiradi.

Java backend sohasida eng kuchli texnologiyalardan biri bo‘lib kelmoqda. Ayniqsa, Spring Framework Java ekotizimining yuragi hisoblanadi. Spring dastlab murakkab, biroq kuchli korporativ ilovalar uchun mo‘ljallangan bo‘lib, unda “dependency injection”, “inversion of control” kabi zamonaviy arxitektura tamoyillari

keng qo‘llaniladi. Spring Boot esa yana bir qadam oldinga borib, Spring asosida tez va sodda tarzda server ilovasini ishga tushirish imkonini yaratgan. Java hamon bank tizimlari, sug‘urta kompaniyalari, yirik davlat tizimlari, millionlab foydalanuvchilarga xizmat ko‘rsatuvchi web xizmatlar uchun eng ishonchli backend texnologiyalardan biri bo‘lib qolmoqda. Buning sababi uning yuqori xavfsizlik talablari, barqarorlik, kuchli statik tipizatsiya va ko‘p yillik tajribaga ega bo‘lgan infratuzilmasidir. Java virtual mashinasi (JVM) orqali kross-platforma imkoniyatini yaratadi, bu esa serverlarni turli operatsion tizimlarda muammosiz ishga tushirishga yordam beradi.



PHP esa backend texnologiyalari ichida eng qadimiylaridan biri bo‘lib, internetning dastlabki rivojlanish bosqichida asosiy rolni o‘ynagan. Bugungi kunda ham PHP ommabop bo‘lishida davom etmoqda, chunki WordPress, Laravel, Symfony kabi gigant ekotizimlar aynan shu tilga asoslangan. Laravel hozirgi zamon PHP dasturlashining eng yorqin namunasi bo‘lib, unda MVC arxitekturasi, qulay routing tizimi, kuchli ORM (Eloquent), migration tizimi, qadoqlangan xavfsizlik mexanizmlari hamda blade templating tizimi mavjud. Laravel o‘zining toza sintaksisi, aniq struktura va qulay yondashuvi bilan dasturchilar orasida juda katta hurmatga ega bo‘lgan. PHP o‘rnatilishi oson, hosting xizmatlari bilan mos va webga maxsus optimallashtirilganligi sababli hanuzgacha millionlab loyihalarda qo‘llanilmoqda.

Backend texnologiyalarini taqqoslaganda, ular o‘z falsafasi, ishlash modeli, qo‘llanilish sohasi va kuchli tomonlariga ko‘ra bir-biridan farq qiladi. Node.js tezkor I/O operatsiyalari talab qiladigan muhitlar uchun qulay bo‘lsa, Pythonning Django freymvorki murakkab tizimlarni tez rivojlantirish uchun eng ideal vosita sifatida

ko‘riladi. Java Spring esa korporativ darajadagi, yuqori ishonchlilikni talab qiladigan tizimlarda ajralmas platformadir. PHP va Laravel esa webga yo‘naltirilgan va dunyoning eng ko‘p ishlatiladigan CMS tizimlari bilan uyg‘un bo‘lganligi sababli hanuzgacha o‘z mavqeini yo‘qotmagan.

Backend texnologiyalarining ustunligi shundaki, ular doimo global talablarga mos ravishda yangilanib boradi. Har bir texnologiya o‘zining ekotizimi, hamjamiyati, arxitektura uslubi, xavfsizlik standartlari va rivojlanish strategiyasiga ega. Shu sababdan backend dasturchi ushbu texnologiyalarning mohiyati, afzalliklari, imkoniyatlari va cheklovlarini chuqur tushunishi, ulardan qaysi biri muayyan loyiha uchun mos kelishini tahlil qila olishi lozim. Texnologiyani to‘g‘ri tanlash nafaqat serverning tezligi va barqarorligini, balki butun tizimning uzoq muddatli rivojlanish salohiyatini belgilab beradi.

5. Amaliy qism: Oddiy “Hello World” server yaratish (Node.js va Python misolida)

Backend dasturlashni chuqur o‘rganishda nazariy bilimlar bilan bir qatorda amaliy tajriba ham katta ahamiyatga ega. Chunki backendning asl mohiyati — mijoz so‘roviga javob beruvchi, serverda ishlaydigan jarayonlarni boshqarish, ma’lumotlarni qayta ishlash va tizimning barqaror ishlashini ta’minlashdir. Shu sababli dasturchining dastlabki qadamlari eng sodda server yaratishdan boshlanadi. Tarixan “Hello World” serveri backend o‘rganishni boshlayotganlar uchun klassik misol bo‘lib kelgan. U sodda bo‘lsa-da, serverning qanday ishlashi, qanday qilib so‘rovni qabul qilishi va mijozga javob qaytarishini amalda ko‘rsatib beradi.

Backend server yaratishda turli texnologiyalar qo‘llanishi mumkin: Node.js, Python, PHP yoki Java. Ammo o‘rganish jarayonida eng sodda va tushunarli yondashuv Node.js yoki Python freymvorklari yordamida server yaratishdan boshlanadi. Chunki bu vositalar minimal kod bilan ishlovchi, tez ishga tushiriladigan va murakkab sozlash talab qilmaydigan muhit yaratadi. Node.js asosan JavaScript tiliga asoslanadi va uning voqea asosidagi (event-driven) arxitekturasi oddiy serverlar

yaratishni juda yengil qiladi. Python esa o‘zining sodda sintaksisi va ixcham freymvorklariga ega bo‘lgani uchun boshlang‘ich darajadagi backend amaliyotlari uchun juda qulay hisoblanadi.

Oddiy server yaratish jarayoni aslida bir necha bosqichdan iborat: avvalo server kutayotgan port belgilanadi, so‘ng HTTP protokoli asosida so‘rov qabul qilinadi, so‘rovni qayta ishlashning sodda qismi bajariladi va yakunda javob qaytariladi. Dasturchi uchun muhim jihat shundaki, bu jarayonda serverning qanday ishlash mexanizmi real ko‘rinishda namoyon bo‘ladi. Masalan, Node.js’da yaratilgan kichik server barcha kelayotgan so‘rovlarga “Hello World” matnini qaytaradi. Bu jarayonda serverning “listen” funksiyasi faollashadi, bu esa uni ko‘rsatilgan portda doimiy kuzatuv holatida ushlab turadi.

Python tilida esa Flask freymvorkidan foydalanib juda ixcham server yaratish mumkin. Flaskning asosiy ustunligi shundaki, u minimalistik arxitekturaga ega bo‘lib, bitta faylda ham serverni ishga tushirish imkonini beradi. Flask orqali yaratilgan serverda dasturning marshrutlash (routing) qismi asosiy o‘rinni egallaydi. Bu yerda mijozning qaysi URL manziliga so‘rov yuborganiga qarab server turli javoblarni yuboradi. “Hello World” misolida esa asosiy URL manzilga kirgan mijozga matn qaytariladi.

Shu sodda amaliy misollar orqali backendning eng muhim jihatlari o‘z aksini topadi: mijoz tomonidan yuborilgan HTTP so‘rovi serverga keladi, server uni qayta ishlaydi, kerakli javobni tayyorlaydi va mijozga qaytaradi. Bu har qanday backend ilovasining yuragi hisoblanadi. Real loyihalarda, albatta, bu jarayon ancha murakkab bo‘ladi — ma’lumotlar bazasiga murojaat qilish, autentifikatsiya, xavfsizlik, ma’lumotlarni formatlash, API yaratish, foydalanuvchi ruxsatlarini tekshirish kabi jarayonlar qo‘shiladi. Ammo poydevor sifatida “Hello World” serverini tushunib olish, serverning qanday ishlashini amaldagi misolda ko‘rish backendning keyingi bosqichlarini osonlashtiradi.

Yakunda shuni aytish joizki, oddiy server yaratish backend dasturchilikka kirishning eng samarali usullaridan biridir. Sababi bu kichik tajriba serverning ichki ishlash tamoyillarini, protokol bilan ishlashni, tarmoq logikasini va kodning bajarilish jarayonini sodda ko‘rinishda namoyon qiladi. Bundan kelib chiqib, Node.js va Python orqali server yaratish faqat amaliy mashq bo‘lib qolmay, balki dasturchining keyingi bilimlarini shakllantiruvchi poydevor ham bo‘lib xizmat qiladi.

Foydalanilgan adabiyotlar

1. Tanenbaum, A. S., & Wetherall, D. **“Computer Networks.”** 5th Edition. Pearson Education, 2011. (Client–server arxitekturasi va tarmoq protokollari bo‘yicha fundamental manba)
2. Fielding, R. T., etal. **RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing.** Internet Engineering Task Force (IETF), 2014. (HTTP strukturasi va xabar formati bo‘yicha rasmiy standart)
3. Fielding, R. T., et al. **RFC 7231: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content.** IETF, 2014.
4. Grinberg, M. **“Flask Web Development.”** O’Reilly Media, 2018. (Python Flask freymvorki yordamida backend yaratish bo‘yicha amaliy qo‘llanma)
5. Django Software Foundation. **Django Documentation.** https://docs.djangoproject.com
   (Django arxitekturasi, ORM, routing va xavfsizlik bo‘yicha rasmiy hujjatlar)
6. Laravel LLC. **Laravel Official Documentation.** https://laravel.com/docs