Szakmai **tudásom** és **ahová** tartok…

|  |  |
| --- | --- |
| Jelenleg | |
| Frontend | HTML |
| CSS |
| JavaScript (Vue, Vite,) |
| Bootstrap |
|  |
|  |
| Backend | **C#** |
| Php |
| Python |
| **SQL** |
| Turso |
| fly.io |
|  |
| Vegyes Extra Tudás | **Docker** |
| **GitHub** |
| **JavaScript** |
| **Dia-Diagram** |
| **Figma** |
|  |

## Python és AI alapú technológiák:

* Mivel az AI és az automatizáció a jövőben is domináns terület lesz, érdemes lehet AI-ra specializálódnod.
* Full-stack fejlesztés: A full-stack fejlesztői szerep mindig keresett, és a Vue.js, Vite és PHP/Python kombinációval nagy eséllyel találsz megfelelő állást.
* Felhőalapú technológiák és DevOps: Fly.io, Docker, és CI/CD rendszerek irányába történő elmélyülés is biztosíthatja a magas keresetet.

Az **AI** **mellett** a full-stack fejlesztés vagy a cloud alapú megoldások kombinációja erős alapot biztosíthat a céljaid eléréséhez.

## AI és gépi tanulás:

### Python:

* Az AI és gépi tanulás szempontjából a Python a legjobb választás. Ha érdekelnek az **AI alapú technológiák**, érdemes lenne olyan **keretrendszereket** megtanulnod, mint a **TensorFlow** vagy **PyTorch**, és esetleg **adatfeldolgozásban** is elmélyedni.
* **AI** és **adatvezérelt** **alkalmazások** iránt is egyre nagyobb a kereslet, és itt magasabb fizetési sávokra is számíthatsz.

|  |
| --- |
| 1. hét: Docker és GitHub alapok |

1. nap – 5 óra: Docker alapok átismétlése

### Első óra – Docker bevezetés

* + **Miért használjuk?**

A Docker gyors és könnyen kezelhető módszert kínál az alkalmazások csomagolására, futtatására és különböző környezetek közötti mozgatására. Különösen hasznos DevOps környezetben, ahol folyamatos integrációra és automatizálásra van szükség.

* + **Miért hasznos a konténerizáció?**

A Docker egy konténerizációs platform, amely lehetővé teszi, hogy alkalmazásokat futtassunk elszigetelt környezetekben (konténerekben). Ennek köszönhetően ugyanaz az alkalmazás minden környezetben (fejlesztés, teszt, éles környezet) azonosan működik. Ez megszünteti az "azon a gépen működött" problémát.

* + **Hivatkozás**: [Docker Bevezető](https://docs.docker.com/get-started/)
  + **Tananyag**:

Áttekintés a Docker CLI parancsokról (pl. docker run, docker ps, docker stop).

* + **Feladat**:

Telepítsd a Dockert, és hozz létre egy egyszerű konténert.

### Második óra – Konténer és image létrehozás

* + **Jegyzetek:**

A Docker image-ek a konténerek alapját képezik, ezek tartalmazzák az alkalmazás futtatásához szükséges fájlokat, könyvtárakat és beállításokat. Egy konténer egy futó példánya az image-nek.

* + **Miért használjuk?:**

Az image-ek lehetővé teszik az alkalmazások könnyű és gyors másolását és futtatását különböző környezetekben, míg a konténerek futó példányok, amelyek könnyen fel- és leállíthatók.

* + **Feladat**:

Hozz létre egy egyszerű Docker image-et egy Node.js alkalmazás számára.

* + **Jegyzetek**:

"Dockerfile felépítése és használata"

* + **Hivatkozás**: [Node.js Docker Alkalmazás Létrehozása](https://nodejs.org/en/guides/nodejs-docker-webapp)
  + **Gyakorlat**: Hozz létre egy saját image-et és futtass egy konténert (Példa feladat 1).

### Harmadik óra – Docker Compose

* + **Jegyzetek:**

A Docker Compose egy eszköz, amellyel több konténeres alkalmazásokat indíthatsz és kezelhetsz egyetlen YAML fájl alapján. Ez lehetővé teszi több konténer egyszerre történő indítását és konfigurálását (pl. egy alkalmazás és egy adatbázis együtt).

* + **Miért használjuk?**

Amikor egy alkalmazás több szolgáltatást használ (pl. adatbázis, cache, backend, frontend), a Docker Compose segítségével ezek egyszerre kezelhetők és könnyen újraindíthatók.

* + **Hivatkozás**: Docker Compose Bevezető
  + **Feladat**: **Több** **konténeres** **alkalmazás** létrehozása Docker **Compose** segítségével (Példa feladat 2).
  + **Jegyzetek**: "docker-compose.yml struktúrája"

### Negyedik óra – Gyakorlat: Komplex alkalmazások konténerizálása

* + **Jegyzetek:**

MySQL konténerizálás lehetősége, ahol egy webalkalmazás adatbázissal együtt fut. A Docker Compose lehetővé teszi, hogy egy YAML fájlban specifikáljuk az összes szükséges szolgáltatást.

* + **Miért használjuk?**

Az alkalmazás különböző részeit (backend, adatbázis, cache) külön konténerekbe helyezve biztosíthatjuk a könnyű skálázhatóságot és konfigurálhatóságot.

* + **Hivatkozás**: Docker + MySQL példa
  + **Példa** **feladat** 3: Egy egyszerű webalkalmazás konténerizálása, adatbázis kapcsolattal (pl. MySQL konténer használata).
  + **Jegyzetek**: "Adatbázis konténerek használata"
  + **Feladat**: Webalkalmazás konténerizálása, MySQL adatbázis konténer használatával.

### Ötödik óra – Összefoglalás és gyakorlati példák

* + **Jegyzetek:**

A konténerizálás előnyei és hátrányai, a Docker Compose használatával létrehozott komplex alkalmazások tesztelése. Jegyzetek frissítése, további gyakorlati feladatok megoldása.

* + **Feladat:**

Teljes alkalmazás konténerizálása, több szolgáltatás konténerizálása Docker Compose segítségével.

Készítsd el és futtasd a saját többkonténeres alkalmazásodat!

2. nap – 5 óra: GitHub és CI/CD bevezetése

### Első óra – Git és GitHub alapok

* + **Jegyzetek:**

A Git verziókezelő rendszer lehetővé teszi a kódverziók nyomon követését, a csapatmunkát és az egyes fejlesztési ágazatok (branch-ek) kezelését. A GitHub tárolja a verziókezelési rendszer fájljait, és lehetővé teszi a közös munkát online.

* + **Miért használjuk?**

Segíti a csapatmunkát, és lehetővé teszi a kód tárolását, felülvizsgálatát és változtatások visszavonását, ami elengedhetetlen a modern fejlesztési munkafolyamatokban.

* + **Feladat**: Repository létrehozása GitHubon, alapvető Git parancsok gyakorlása.
  + **Jegyzetek**: "Git parancsok összefoglalója" (commit, push, pull, merge).

### Második óra – GitHub Actions bevezetés

* + **Jegyzetek**

A GitHub Actions segítségével CI/CD pipeline-okat hozhatsz létre, amelyek automatikusan tesztelik, buildelik és deployolják az alkalmazásodat minden új commit után.

* + **Miért használjuk?**

Az automatizált folyamatok gyorsabbá és megbízhatóbbá teszik az alkalmazás fejlesztési és telepítési ciklusait.

* + Hivatkozás: [GitHub Actions Bevezető](https://docs.github.com/en/actions)
  + Feladat: Egyszerű workflow létrehozása, amely buildeli az alkalmazást (Példa feladat 1).
  + Jegyzetek: "GitHub Actions workflow szintaxis"

### Harmadik óra – CI/CD pipeline mélyítése

* + **Jegyzetek:**

A CI/CD (Continuous Integration/Continuous Deployment) biztosítja, hogy minden kódváltoztatás automatikusan tesztelve és deployolva legyen, minimalizálva az emberi hibák lehetőségét.

* + **Miért használjuk?**

A fejlesztés során a kód automatikus tesztelése és telepítése csökkenti a hibák esélyét, és felgyorsítja a fejlesztési folyamatot.

* + **Hivatkozás:** [CI/CD bevezető](https://aws.amazon.com/devops/continuous-integration/)
  + **Komplexebb pipeline építése:** Tesztelési folyamatok beépítése (Példa feladat 2: Egyszerű unit teszt futtatása).
  + Jegyzetek: "CI/CD folyamatok felépítése"

### Negyedik óra – Deployment pipeline építése

* + **Jegyzetek:**

A deployment pipeline segítségével a kód automatikusan települ az előre meghatározott környezetbe (teszt, éles környezet). Docker alapú alkalmazásoknál ez a konténerek telepítését jelenti.

* + **Miért használjuk?**

Ezáltal gyorsabb és biztonságosabb lesz a kód telepítése különböző környezetekbe.

* + **Automatizált** **deploy** **pipeline** **létrehozása** Docker konténerekhez.
  + Feladat: Docker alapú alkalmazás automatizált deploy pipeline-ja (Példa feladat 3).
  + Jegyzetek: "Deployment pipeline lépései"

### Ötödik óra – Összefoglalás és dokumentáció

* + **Jegyzetek:**

A CI/CD pipeline létrehozásának teljes dokumentálása, optimalizálás és további funkciók hozzáadása.

* + **Feladatok:**

Hozz létre egy teljes CI/CD folyamatot, és telepíts egy alkalmazást automatizáltan.

* + Tananyag:

CI/CD legjobb gyakorlatok összegzése. Jegyzetek frissítése, a CI/CD folyamat dokumentálása.

|  |
| --- |
| 2. hét: AI és gépi tanulás alapjai - *AI/ML Alapjai* - |

1. nap – 5 óra: Python alapok és gépi tanulás könyvtárak

### Első óra – Python ismétlés

* + Jegyzetek:

A Python egy sokoldalú és könnyen tanulható programozási nyelv, amely különösen népszerű az adatfeldolgozás, gépi tanulás és webfejlesztés terén. Ismételd át a Python alapvető szintaxisát (változók, ciklusok, függvények), mivel ezek fontosak lesznek a későbbi gyakorlatokhoz.

* + Miért használjuk?

A Python rugalmas és erőteljes nyelv, amelyet könnyű megtanulni, és számos területen alkalmazható, különösen adatelemzésre és gépi tanulásra.

* + Fontos alapok:

Változók és adattípusok (számok, karakterláncok, listák, szótárak)

Ciklusok és feltételes utasítások (for, while, if)

Függvények és osztályok

* + Gyakorlati feladat:

Írj egy Python programot, amely összegzi egy lista elemeit, majd egy másik függvénnyel rendezi a listát.

* + Hivatkozás: [Suilpy.hu](https://sulipy.hu)
  + Feladat:

Python alapvető szintaxis áttekintése (pl. változók, ciklusok, függvények).

* + Jegyzetek:

"Python alapok gyors összefoglaló"

### Második óra – Adatfeldolgozás Pandas és NumPy használatával

* + Jegyzetek:

A Pandas egy adatfeldolgozó könyvtár, amely DataFrame-eket használ adatok rendezésére és elemzésére. A NumPy lehetővé teszi a matematikai műveletek és mátrixszámítások elvégzését hatékonyan.

* + **Miért használjuk?**

Az adatelemzés elengedhetetlen része a gépi tanulásnak, és ezek az eszközök lehetővé teszik, hogy hatékonyan és gyorsan dolgozzunk nagy mennyiségű adattal.

* + **Fontos alapok:**
    - * + ***Pandas DataFrame:***

Hogyan lehet létrehozni, adatokat betölteni (pl. CSV-ből), és alapvető műveleteket végezni (filtelés, rendezés, csoportosítás).

* + - * + ***NumPy tömbök:***

Hogyan lehet tömböket létrehozni és matematikai műveleteket végezni rajtuk.**Feladat**:

Egyszerű adatfeldolgozási műveletek végrehajtása (pl. CSV fájlok beolvasása, adatok tisztítása).

* + Gyakorlati feladat:

Tölts be egy CSV fájlt Pandas segítségével, majd elemezd az adatokat (pl. átlag, maximum, minimum értékek kiszámítása).

* + Hivatkozás:

*Pandas Dokumentáció*

*NumPy Dokumentáció*

### Harmadik óra – Gépi tanulás bevezetés

* + Jegyzetek:

A gépi tanulás lehetővé teszi az algoritmusok számára, hogy tanuljanak az adatokból, azonosítsanak mintázatokat, és ezek alapján előrejelzéseket tegyenek. A scikit-learn egy jól ismert gépi tanulási könyvtár Pythonban, amely egyszerű, jól dokumentált algoritmusokat biztosít különböző típusú problémák megoldására (pl. regresszió, osztályozás, klaszterezés).

* + Miért használjuk?

A regressziós modellek segítenek az összefüggések feltárásában két vagy több változó között, ami hasznos lehet a pénzügyi előrejelzésektől az egészségügyi adatelemzésig.

* + Feladat:

Építs egy lineáris regressziós modellt, amely megjósolja a házárakat különböző változók alapján (pl. szobák száma, méret).Jegyzetek: "Gépi tanulási algoritmusok áttekintése"

Hivatkozás: Scikit-learn Lineáris Regresszió

### Negyedik óra – Gyakorlati feladat

* + Jegyzetek:

A döntési fák egy alapvető, de erős algoritmus, amely könnyen magyarázható és vizualizálható. Ez egy rekurzív felosztáson alapuló algoritmus, amely elágazásokkal dönt a bemeneti adatok kategorizálásáról.

* + Miért használjuk?

A különböző modellek összehasonlítása segít kiválasztani a legjobb teljesítményű algoritmust az adott problémára.

* + Példa feladat:

Egyszerű gépi tanulási modell építése és tesztelése (pl. döntési fa használata). Próbálj ki különböző gépi tanulási modelleket (pl. kNN, SVM), és hasonlítsd össze az eredményeket.

* + Jegyzetek: "Modellek tréningelése és validálása"
  + Hivatkozás: *Scikit-learn Lineáris Regresszió*

### Ötödik óra – Összefoglalás és jegyzetek frissítése

* + Gyakorlatok áttekintése, jegyzetek összefoglalása.
  + Feladat: További feladatok megoldása Python és gépi tanulás terén.

1. Hét – 2. Nap (5 óra): Haladó gépi tanulás témák

### Első óra – Neurális hálózatok bevezetése

* + Jegyzetek:

A neurális hálózatok mesterséges neuronokból állnak, és képesek komplex mintázatok megtanulására. A TensorFlow és Keras könyvtárak könnyen kezelhető eszközöket biztosítanak a neurális hálózatok építésére és tréningelésére.

* + Miért használjuk?

A neurális hálózatok képesek komplex mintázatokat tanulni, és nagyszerűen alkalmazhatók olyan feladatokra, mint a képfelismerés vagy a természetes nyelvfeldolgozás.

* + Feladat:

Építs egy egyszerű, egy rétegű neurális hálózatot a Keras segítségével, amely osztályozási problémákat old meg.

Hivatkozás: Keras Bevezető

### Második óra – Modell optimalizálás

* + Jegyzetek:

Az optimalizálási technikák lehetővé teszik a modellek pontosabbá és gyorsabbá tételét. A leggyakoribb módszerek közé tartozik a tanulási sebesség beállítása, a régularizáció hozzáadása és a rétegek számának módosítása.

* + Miért használjuk?

Az optimalizált modellek jobb pontosságot és hatékonyságot biztosítanak, különösen nagyobb adathalmazokon.

* + Feladat:

Optimalizálj egy neurális hálózatot a tanulási sebesség, rétegszám és aktivációs függvények finomhangolásával.

Hivatkozás: Keras Hiperparaméter Tuning

### Harmadik óra – Adatvizualizáció

* + Jegyzetek:

Az adatvizualizáció lehetővé teszi az adatok és a modellek teljesítményének szemléletes bemutatását. A matplotlib és seaborn könyvtárak segítségével könnyen készíthetsz grafikonokat és hőtérképeket.

* + Miért használjuk?

A vizualizáció segíti a döntéshozatalt és a modellek kiértékelését, ami kulcsfontosságú a fejlesztési folyamat során.

* + Feladat:

Vizualizálj egy gépi tanulási modellt és annak teljesítményét különböző metrikák alapján (pl. ROC görbe, precízió-recall).

*Hivatkozás: matplotlib Dokumentáció*

### Negyedik óra – Modellek kiértékelése

* + Jegyzetek:

A modellek kiértékelése különböző metrikák alapján történik, mint például az MSE (Mean Squared Error), precízió, recall vagy az F1-mutató. Ezek a metrikák segítenek megérteni, hogy a modell mennyire jól működik az új adatokon.

* + Miért használjuk?

A kiértékelési módszerek segítenek megérteni a modell általánosítási képességét, és javítják a teljesítményét.

* + Feladat:

Értékeld ki egy gépi tanulási modell teljesítményét különböző adathalmazokon, és hasonlítsd össze a különböző metrikák alapján.

* + Hivatkozás: Modellek Értékelése

### Ötödik óra – Projekt összefoglalása és dokumentálás

* + Jegyzetek:

A teljes tanulási folyamat összefoglalása és a kész modellek dokumentálása fontos lépés, hogy később visszatérhess hozzá, vagy megoszthasd másokkal a munkádat.

* + Miért használjuk?

A megfelelő dokumentáció segít megérteni a projekt minden lépését, és visszakereshetővé teszi a fontos döntéseket és eredményeket.

* + Feladat:

Dokumentáld a neuronháló projekted, beleértve a modell építésének és optimalizálásának lépéseit, valamint a kiértékelés eredményeit.

* + Hivatkozás: [*TensorFlow Projekt Dokumentáció*](https://www.tensorflow.org/)

|  |
| --- |
| 3. hét: DevOps haladó témák |

1. nap – 5 óra: Kubernetes bevezetés

### Első óra – Kubernetes alapok

* + Jegyzetek:

A Kubernetes egy nyílt forráskódú konténer-orchestration rendszer, amely automatizálja a konténerek telepítését, kezelését és skálázását. A Minikube egy olyan eszköz, amely lehetővé teszi a Kubernetes telepítését és futtatását helyi gépeken.

* + Miért használjuk?

A Kubernetes segítségével kezelhetjük a komplex, többkonténeres alkalmazásokat, lehetővé téve az automatikus skálázást és deployolást.

* + Feladat:

Telepítsd a Minikube-ot, és állíts fel egy Kubernetes clustert a helyi gépeden.

* + Hivatkozás: [Minikube Telepítés](https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download)

### Második óra – Podok és szolgáltatások kezelése

* + Jegyzetek:

A pod a Kubernetes legkisebb futtatható egysége, amely egy vagy több konténert tartalmaz. A podok állapotát és elérhetőségét a Kubernetes kezeli. A deployment pedig a podok frissítését és skálázását teszi lehetővé.

* + Miért használjuk?

A podok segítségével a konténerek könnyen skálázhatók, valamint automatizálható a frissítés és újratelepítés.

* + Feladat:

Hozz létre egy egyszerű alkalmazást (pl. egy Node.js app) egy podban, és futtasd Kubernetes környezetben.

* + Hivatkozás: [Podok és deploymentek kezelése](https://azure.microsoft.com/hu-hu/solutions/kubernetes-on-azure/get-started).

### Harmadik óra – Kubernetes konfiguráció

* + Jegyzetek:

A Kubernetes konfigurációs fájlokat (YAML) használ arra, hogy leírja az alkalmazások összetevőit, például a podokat, deploymenteket és service-eket. A YAML fájlok lehetővé teszik a Kubernetes alkalmazások deklaratív módon történő definiálását.

* + Miért használjuk?

A YAML fájlokkal könnyen újra felhasználhatók a konfigurációk, és automatizálhatók a deploy folyamatok.

* + Feladat:

Írj egy YAML fájlt, amely tartalmaz egy deploymentet és egy service-t, majd deployold a Kubernetes clusterre.

Hivatkozás: [Kubernetes YAML Konfiguráció](https://learn.microsoft.com/hu-hu/azure/copilot/generate-kubernetes-yaml)

### Negyedik óra – Gyakorlat: Kubernetes alkalmazás

* + Jegyzetek:

Egy összetett alkalmazás futtatásához a Kubernetes különböző szolgáltatásokat használ, mint például volume-ok (állandó tárolási megoldások) és service-ek, amelyek az alkalmazások közötti kommunikációt kezelik.

* + Miért használjuk?

A Kubernetes lehetővé teszi az összetett alkalmazások könnyű telepítését, amelyek több szolgáltatást használnak egyszerre, például adatbázisokat és webszolgáltatásokat.

* + **Példa feladat:**

Hozz létre egy több konténeres alkalmazást (pl. egy webalkalmazást adatbázissal) és deployold a Kubernetes clusterre.

* + Jegyzetek: [Kubernetes szolgáltatások és volume-ok kezelése](https://www.shiwaforce.com/blog/devops/2018/10/18/kubernetes-3-ora-alatt-reszletes-utmutato-3-resz-kubernetes/)

### Ötödik óra – Összefoglalás és gyakorlás

* + Jegyzetek:

A nap végén fontos összefoglalni, hogy hogyan működik a Kubernetes a különböző komponensek között (podok, deploymentek, service-ek, volume-ok).

* + Miért használjuk?

Az összetett alkalmazások kezelése és optimalizálása Kubernetes alatt megmutatja, hogyan lehet skálázni és menedzselni a valós alkalmazásokat különböző környezetekben.

* + Feladat:

Optimalizáld és deployold a létrehozott alkalmazást, és dokumentáld a folyamatot.

* + Hivatkozás: [Kubernetes alapok](https://www.shiwaforce.com/blog/devops/2018/05/24/kubernetes-3-ora-alatt-kiindulas-mikroszervizes-alkalmazasokbol/)

1. Hét 2. nap – 5 óra: Haladó Kubernetes és DevOps témák

### Első óra – Kubernetes skálázás

* + ****Jegyzetek****:

A Kubernetes automatikus skálázási funkciói lehetővé teszik, hogy a podok számát dinamikusan növeljük vagy csökkentsük a terhelés alapján. A **Horizontal Pod Autoscaler (HPA)** automatikusan méretezi a podokat a CPU vagy memória használatának függvényében.

* + ****Miért használjuk?****

A skálázás lehetővé teszi, hogy hatékonyan kezeljük a forgalom ingadozásait és erőforrásokat takarítsunk meg, amikor nincs szükség túl sok podra.

* + ****Feladat****:

Konfigurálj egy HPA-t egy egyszerű alkalmazásra, és figyeld meg, hogyan méretezi a podokat a terhelés növekedésekor.

* + ****Hivatkozás****: [Kubernetes Horizontal Pod Autoscaler](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/)

### Második óra – Kubernetes monitoring

* + **Jegyzetek**:

A Kubernetes monitoring nélkülözhetetlen ahhoz, hogy lássuk, hogyan teljesít az alkalmazás a clusterben. Az olyan eszközök, mint a **Prometheus** és a **Grafana** segítenek figyelni a podok erőforrás-felhasználását és teljesítményét.

* + ****Miért használjuk?****

A megfelelő monitoring lehetővé teszi az alkalmazások teljesítményének nyomon követését és a hibák gyors felismerését.

* + ****Feladat****:

Telepítsd a Prometheus-t és a Grafana-t a clusteredre, és monitorozd az alkalmazásod erőforrás-használatát.

* + ****Hivatkozás****: [Prometheus Monitoring](https://prometheus.io/)

### Harmadik óra – CI/CD pipeline Kubernetes alatt

* + ****Jegyzetek****:

A Kubernetes-t integrálhatjuk a folyamatos integrációs és telepítési (CI/CD) folyamatokba, hogy az alkalmazások automatikusan deployolódjanak a Kubernetes clusterbe, miután a kód frissült. A **Jenkins**, **GitLab CI** vagy **GitHub Actions** segítségével hozhatunk létre automatizált pipeline-okat.

* + ****Miért használjuk?****

Az automatizált deploy folyamatok lehetővé teszik a gyorsabb fejlesztési ciklusokat, csökkentik az emberi hibák esélyét, és növelik a termelékenységet.

* + ****Feladat****:

Hozz létre egy CI/CD pipeline-t, amely automatikusan deployolja az alkalmazást a Kubernetes clusterre.

* + ****Hivatkozás****: [Jenkins Kubernetes CI/CD](https://medium.com/@jaiymzndubuisi/a-step-by-step-guide-to-ci-cd-with-jenkins-github-and-kubernetes-7b8090146a07)

### Negyedik óra – Kubernetes security

* + ****Jegyzetek****:

A Kubernetes biztonságos használata kritikus fontosságú, különösen az érzékeny adatokat kezelő alkalmazások esetén. Fontos megérteni a podok izolációját, a jogosultságok kezelését és a titkosítási mechanizmusokat.

* + ****Miért használjuk?****

A biztonság biztosítja, hogy a Kubernetes környezetek védettek legyenek a támadásokkal és adatszivárgásokkal szemben

* + ****Feladat****:

Alkalmazd a Kubernetes biztonsági gyakorlatokat, mint például a **Role-Based Access Control (RBAC)** és a titkos kulcsok használata.

* + ****Hivatkozás****: [Kubernetes Biztonság](https://nki.gov.hu/it-biztonsag/hirek/kubernetes-biztonsagi-iranyelvek-az-nsa-tol-es-a-cisa-tol/).

### Ötödik óra – Összefoglalás és dokumentálás

* + ****Jegyzetek****:

A nap során tanultak összegzése és a Kubernetes skálázási, monitoring, CI/CD és biztonsági funkcióinak dokumentálása.

* + ****Miért használjuk?****

A jól dokumentált folyamatok és a rendszeres visszatekintés lehetővé teszi a Kubernetes hatékony használatának áttekintését és optimalizálását.

* + ****Feladat****:

Dokumentáld a teljes folyamatot a Kubernetes alapjaitól a haladó DevOps gyakorlatokig, és mutasd be, hogyan integráltad ezeket a valós alkalmazásokba.

* + ****Hivatkozás****: [Kubernetes dokumentáció](https://azure.microsoft.com/hu-hu/resources/cloud-computing-dictionary/what-is-kubernetes#n%C3%A9zze-meg,-hogyan-m%C5%B1k%C3%B6dik-a-kubernetes).

További hetek kidolgozása:

* **Hónapokra bontva** haladnánk tovább mindkét fő irányban (DevOps + AI).
* Minden hét végén összefoglaló jegyzetek és további feladatok segítenének a gyakorlásban.

# ****1. Hónap: Alapok lerakása****

## **1-2. hét: Git és verziókezelés (10-20 óra)**

* + ****Célok:****

Git alapok elsajátítása

GitHub használata projektek tárolására és megosztására

* + ****Tananyag:****

Git alapvető parancsok (init, add, commit, push, pull, clone)

Branch-kezelés és merge konfliktusok megoldása

GitHub használata, repository-k létrehozása és kezelése

Gyakorlás saját projektek verziókezelésével

* + ****Források:****

[*Pro Git könyv (magyarul)*](https://git-scm.com/book/hu/v2)

[*GitHub Learning Lab*](https://lab.github.com/)

## **3. hét: Projektmunka – Git alapú projekt létrehozása**

* + ****Feladat:****

Készíts egy egyszerű alkalmazást (pl. "Hello World" webalkalmazás)

Alkalmazd a Git verziókezelést a projekt során

Használd a GitHub-ot a kód tárolására és megosztására

Dokumentáld a projektet a README.md fájlban

### ****4. hét: Linux alapok és parancssori ismeretek (5-10 óra)****

* **Célok:**
  + Alapvető Linux parancsok elsajátítása
  + Shell scriptelés alapjai
* **Tananyag:**
  + Navigáció a fájlrendszerben (cd, ls, pwd)
  + Fájlok és könyvtárak kezelése (cp, mv, rm, mkdir)
  + Jogosultságok és engedélyek kezelése (chmod, chown)
  + Alapvető shell scriptek írása
* **Források:**
  + Linux parancssori alapok
  + Bash scripting útmutató

## **2. Hónap: Konténerizáció Dockerrel**

### ****5-6. hét: Docker alapok (10-20 óra)****

* **Célok:**
  + Megérteni a Docker működését és előnyeit
  + Saját Docker image-ek és konténerek létrehozása
* **Tananyag:**
  + Docker telepítése és beállítása
  + Alapvető Docker parancsok (docker run, docker build, docker images, docker ps)
  + Dockerfile írása és testreszabása
  + Hivatalos Docker image-ek használata
* **Források:**
  + Docker Dokumentáció
  + [Docker Tutorial magyarul](https://dockerdocs.hu/)

### ****7. hét: Projektmunka – Egyszerű alkalmazás konténerizálása****

* **Feladat:**
  + Válassz egy egyszerű webalkalmazást (pl. Node.js, Python Flask)
  + Írj egy Dockerfile-t az alkalmazáshoz
  + Építsd meg az image-et és futtasd konténerben
  + Töltsd fel a projektet a GitHub-ra, dokumentációval együtt

### ****8. hét: Docker Compose és többkonténeres alkalmazások (5-10 óra)****

* **Célok:**
  + Többkonténeres alkalmazások kezelése Docker Compose segítségével
* **Tananyag:**
  + Docker Compose telepítése és használata
  + docker-compose.yml fájl felépítése
  + Szolgáltatások összekapcsolása (pl. webalkalmazás és adatbázis)
  + Környezeti változók és konfiguráció kezelése
* **Források:**
  + Docker Compose Dokumentáció
  + Docker Compose útmutató

## **3. Hónap: CI/CD alapok**

### ****9-10. hét: CI/CD bevezetése és GitHub Actions (10-20 óra)****

* **Célok:**
  + Megérteni a CI/CD fogalmát és fontosságát
  + Automatizált folyamatok létrehozása GitHub Actions segítségével
* **Tananyag:**
  + CI/CD alapelvek és folyamatok
  + GitHub Actions workflow-k létrehozása
  + Egyszerű build és tesztelési folyamatok automatizálása
  + Workflow-k konfigurálása és hibakeresése
* **Források:**
  + [GitHub Actions Dokumentáció](https://docs.github.com/en/actions)
  + CI/CD bevezető

### ****11. hét: Projektmunka – CI/CD pipeline létrehozása****

* **Feladat:**
  + Hozz létre egy CI/CD pipeline-t a korábbi projektedhez
  + Automatizáld a build, tesztelés és deploy folyamatokat
  + Használd a GitHub Actions-t a pipeline kezeléséhez
  + Dokumentáld a folyamatot és frissítsd a projekted a GitHub-on

### ****12. hét: Tesztelés és minőségbiztosítás (5-10 óra)****

* **Célok:**
  + Tesztelési módszerek és eszközök megismerése
  + Automatizált tesztek integrálása a CI/CD folyamatba
* **Tananyag:**
  + Unit tesztek írása (pl. Jest, PyTest)
  + Integrációs tesztek alapjai
  + Tesztlefedettség és minőségellenőrzés
  + Hibakezelés és logolás
* **Források:**
  + Unit tesztelés Node.js-ben
  + [Python Unit Tesztelés](https://docs.python.org/3/library/unittest.html)

## **4-5. Hónap: Kubernetes alapok**

### ****13-14. hét: Kubernetes bevezetése (10-20 óra)****

* **Célok:**
  + Megismerni a Kubernetes alapfogalmait és működését
  + Egyszerű alkalmazások futtatása Kubernetes clusterben
* **Tananyag:**
  + Kubernetes architektúra (master node, worker node, pod, deployment, service)
  + Minikube telepítése és használata helyi fejlesztéshez
  + Alapvető kubectl parancsok
  + YAML fájlok írása és használata Kubernetes konfigurációhoz
* **Források:**
  + Kubernetes Dokumentáció
  + Kubernetes Tutorial

### ****15. hét: Projektmunka – Alkalmazás deployolása Kubernetes-be****

* **Feladat:**
  + Konténerizáld az alkalmazásodat, ha még nem tetted meg
  + Írj Kubernetes konfigurációs fájlokat (deployment, service)
  + Deployold az alkalmazást egy helyi Kubernetes clusterbe (Minikube)
  + Dokumentáld a folyamatot és frissítsd a GitHub projektet

### ****16-17. hét: Kubernetes mélyítése (10-20 óra)****

* **Célok:**
  + Haladó Kubernetes funkciók megismerése
  + Skálázás, frissítés és monitorozás
* **Tananyag:**
  + Skálázás (Horizontal Pod Autoscaler)
  + Rolling updates és rollback-ek
  + Liveness és readiness probe-ok használata
  + Beépített monitorozási eszközök (pl. Kubernetes Dashboard)
* **Források:**
  + Kubernetes Skálázás
  + Monitorozás Kubernetes-ben

### ****18. hét: Projektmunka – Kubernetes skálázás és frissítés****

* **Feladat:**
  + Alkalmazd a skálázási és frissítési technikákat a korábbi Kubernetes projekteden
  + Implementálj liveness és readiness probe-okat
  + Készíts dokumentációt és frissítsd a GitHub projektet

## **6. Hónap: Felhőszolgáltatók alapjai**

### ****19-20. hét: AWS alapok (10-20 óra)****

* **Célok:**
  + Megismerni az Amazon Web Services (AWS) alapvető szolgáltatásait
  + Alkalmazások deployolása AWS környezetbe
* **Tananyag:**
  + AWS fiók létrehozása és biztonsági beállítások
  + EC2, S3, IAM szolgáltatások alapjai
  + AWS CLI használata
  + Egyszerű alkalmazás futtatása EC2 instance-on
* **Források:**
  + [AWS Dokumentáció](https://aws.amazon.com/documentation/)
  + [AWS Ingyenes kurzusok](https://aws.amazon.com/training/)

### ****21. hét: Projektmunka – Alkalmazás deployolása AWS-re****

* **Feladat:**
  + Deployold az alkalmazásodat egy AWS EC2 instance-ra
  + Használd az AWS szolgáltatásait az alkalmazás támogatásához (pl. S3 tárolás)
  + Dokumentáld a folyamatot és frissítsd a GitHub projektet

### ****22. hét: Infrastruktúra mint kód (IaC) – Terraform bevezetése (5-10 óra)****

* **Célok:**
  + Infrastruktúra automatizálása kóddal a Terraform segítségével
* **Tananyag:**
  + Terraform alapok és telepítés
  + Alapvető Terraform konfigurációk írása
  + AWS erőforrások létrehozása Terraformmal
  + Állapotkezelés és modulok használata
* **Források:**
  + Terraform Dokumentáció
  + Terraform AWS útmutató

### ****23. hét: Projektmunka – Infrastruktúra automatizálása Terraformmal****

* **Feladat:**
  + Írj Terraform konfigurációkat az AWS erőforrásaidhoz
  + Automatizáld az alkalmazásod infrastruktúrájának létrehozását
  + Dokumentáld a folyamatot és frissítsd a GitHub projektet

## **7-8. Hónap: Haladó DevOps témák**

### ****24-25. hét: CI/CD pipeline felhőben (10-20 óra)****

* **Célok:**
  + CI/CD folyamatok implementálása felhőalapú szolgáltatásokkal
  + Jenkins vagy más CI/CD eszköz használata
* **Tananyag:**
  + Jenkins telepítése és konfigurálása
  + Pipeline szkriptek írása
  + Integráció felhőszolgáltatásokkal (pl. AWS, Azure)
  + Biztonsági és jogosultsági beállítások
* **Források:**
  + Jenkins Dokumentáció
  + [CI/CD a felhőben](https://aws.amazon.com/devops/continuous-delivery/)

### ****26. hét: Projektmunka – Haladó CI/CD pipeline létrehozása****

* **Feladat:**
  + Készíts egy teljes CI/CD pipeline-t, amely buildeli, teszteli és deployolja az alkalmazásodat az AWS-re
  + Használd a Jenkins-t vagy más eszközt a folyamat automatizálására
  + Dokumentáld a pipeline felépítését és működését

### ****27-28. hét: Monitoring és logolás (10-20 óra)****

* **Célok:**
  + Alkalmazások és infrastruktúra monitorozása
  + Logok gyűjtése és elemzése
* **Tananyag:**
  + Prometheus és Grafana bevezetése
  + Alkalmazások monitorozása Kubernetes-ben
  + Logok kezelése és elemzése (ELK Stack: Elasticsearch, Logstash, Kibana)
  + Riasztások és értesítések beállítása
* **Források:**
  + Prometheus Dokumentáció
  + [ELK Stack bevezető](https://www.elastic.co/what-is/elk-stack)

### ****29. hét: Projektmunka – Monitoring rendszer implementálása****

* **Feladat:**
  + Implementálj egy monitorozási rendszert az alkalmazásodhoz
  + Állíts be riasztásokat és értesítéseket bizonyos eseményekre
  + Dokumentáld a monitorozási megoldásodat és frissítsd a GitHub projektet

## **9-10. Hónap: Biztonság és hálózatok**

### ****30-31. hét: DevOps biztonság (DevSecOps) (10-20 óra)****

* **Célok:**
  + Biztonsági gyakorlatok integrálása a DevOps folyamatokba
  + Alkalmazások és infrastruktúra védelme
* **Tananyag:**
  + Alapvető biztonsági fogalmak és legjobb gyakorlatok
  + Konténerek biztonsága (image scanning, sebezhetőségek kezelése)
  + Kubernetes biztonság (RBAC, hálózati policy-k)
  + Titkos adatok kezelése (Secrets menedzsment)
* **Források:**
  + DevSecOps útmutató
  + Kubernetes Biztonság

### ****32. hét: Projektmunka – Biztonsági megoldások implementálása****

* **Feladat:**
  + Implementálj biztonsági intézkedéseket az alkalmazásod és infrastruktúrád védelmére
  + Végezz sebezhetőség vizsgálatot és javítsd a talált problémákat
  + Dokumentáld a biztonsági megoldásokat és frissítsd a GitHub projektet

### ****33-34. hét: Hálózatok és infrastruktúra (10-20 óra)****

* **Célok:**
  + Hálózati alapok megértése
  + Infrastruktúra tervezés és optimalizálás
* **Tananyag:**
  + TCP/IP protokollok és hálózati modellek
  + Hálózati konfigurációk Kubernetes-ben és felhőszolgáltatásokban
  + Terheléselosztás és skálázhatóság
  + Infrastruktúra optimalizálása költség és teljesítmény szempontjából
* **Források:**
  + Hálózati alapok
  + Kubernetes Hálózatok

### ****35. hét: Projektmunka – Hálózati konfiguráció és optimalizálás****

* **Feladat:**
  + Optimalizáld az alkalmazásod hálózati beállításait
  + Implementálj terheléselosztást és skálázási megoldásokat
  + Dokumentáld a hálózati konfigurációkat és frissítsd a GitHub projektet

## **11-12. Hónap: Összefoglalás és portfólió építés**

### ****36-38. hét: Nagyobb projekt kivitelezése (15-30 óra)****

* **Célok:**
  + Összefoglalni és alkalmazni a tanultakat egy komplex projektben
  + Teljes körű DevOps folyamat implementálása
* **Feladat:**
  + Tervezd meg és valósítsd meg egy összetett alkalmazás teljes infrastruktúráját
  + Használd az összes tanult eszközt és módszert (Docker, Kubernetes, CI/CD, felhőszolgáltatók, biztonság, monitorozás)
  + Dokumentáld részletesen a projektet, beleértve a döntéseket és megvalósításokat

### ****39-40. hét: Portfólió és önéletrajz készítése (5-10 óra)****

* **Célok:**
  + Portfólió összeállítása a GitHub projektekből
  + Versenyképes önéletrajz és motivációs levél írása
* **Feladat:**
  + Rendezett és áttekinthető portfólió készítése
  + Önéletrajz frissítése a megszerzett tudással és projektekkel
  + Készíts egy LinkedIn profilt vagy frissítsd a meglévőt
  + Készülj fel az állásinterjúkra (gyakori kérdések, interjú technikák)
* **Források:**
  + Önéletrajz írási tippek
  + Állásinterjú tippek

### ****41-48. hét: Álláskeresés és folyamatos tanulás****

* **Célok:**
  + Megpályázni a kívánt pozíciókat
  + Továbbfejleszteni a tudást és naprakészen maradni
* **Feladatok:**
  + Álláshirdetések keresése és jelentkezés
  + Interjúk gyakorlása és részvétel az állásinterjúkon
  + Folyamatosan frissíteni a tudást új technológiákkal és trendekkel
  + Hálózatépítés szakmai rendezvényeken és online platformokon
* **Források:**
  + [Profession.hu](https://www.profession.hu/)
  + [LinkedIn](https://www.linkedin.com/)
  + [Meetup csoportok](https://www.meetup.com/)

### ****Összegzés****

Ezzel a módosított tanulási tervvel reális időkeretben, heti 5-10 órás tanulással alapos gyakorlati tudást szerezhetsz a DevOps területén. A projektmunkák segítenek a tanultak alkalmazásában és a portfóliód építésében, ami rendkívül fontos a munkaerőpiacon.

**Tippek a sikeres tanuláshoz:**

* **Rendszeresség:** Tartsd magad a tanulási tervhez, és próbálj meg minden héten időt szánni a tanulásra.
* **Gyakorlatias megközelítés:** Mindig próbáld ki a tanultakat gyakorlatban, ne csak elméletben ismerd meg az eszközöket és technológiákat.
* **Kérdezz és kapcsolódj:** Csatlakozz online közösségekhez, fórumokhoz, ahol kérdéseket tehetsz fel és tanulhatsz másoktól.
* **Dokumentáció:** Minden projektedet dokumentáld részletesen, ez segít a megértésben és a későbbi visszatekintésben.

**Sok sikert kívánok a tanuláshoz és a karriered építéséhez!** Ha bármilyen kérdésed van vagy segítségre van szükséged a tanulási folyamat során, szívesen állok rendelkezésedre.

# Product Owner - Specifikátor/Üzleti Elemző (MySQL tudással)

## Feladatok

* Üzleti igények és visszajelzések gyűjtése a termékfejlesztéshez, beleértve interjúk, UAT-ok, és megbeszélések szervezését és vezetését
* Aktív részvétel az üzleti megoldások tervezésében
* Az igények dokumentálása víziók, roadmap, üzleti folyamatábrák, use case diagramok, user story-k, drótvázak és glosszárium formájában
* Szóbeli egyeztetések fejlesztőkkel és tesztelőkkel
* Release tervezése és dokumentálása (change logok és oktató anyagok készítése a változásokról)
* Üzleti demók szervezése és vezetése
* Kapcsolattartás társosztályokkal, fejlesztéssel, tesztelőkkel
* A termék sikerességének monitorozása és visszajelzések gyűjtése
* A termék napi működésének biztosítása - felmerült problémák kivizsgálása megoldása
* Support szakmai támogatása a termékkel felmerült kérdésekben
* Belső üzleti folyamatok megismerése és megértése
* User guide készítése és karbantartása

## Elvárások

* **Product Owner vagy Business Analyst pozícióban szerzett 2+ éves tapasztalat**
* Agilis szemléletű csapatban szerzett gyakorlat
* Folyamat- és rendszerszemlélet
* Feladatkezelő rendszerek, különösen Jira használatában szerzett tapasztalat
* Bitbucket/GitHub ismerete
* Haladó szintű Excel ismeretek (adatkezelés, elemzés, vizualizálás)
* MySQL adatbázis kezelési ismeretek
* Kiváló kommunikációs képességek az üzleti és fejlesztői oldallal egyaránt
* Önállóság, proaktivitás és jó szervezőkészség

## Előnyt jelent

* Scrum Master, szoftverfejlesztői vagy tesztelői tapasztalat
* Fejlesztési projektekben szerzett tapasztalat
* szoftverfejlesztési módszertanok, kiváltképp az agilis módszertan gyakorlati ismerete

# Webfejlesztő(Budapest)

### Feladataid lennének:

* Új publikus és zárt felhasználású webes alkalmazások tervezése és
* fejlesztése, integ-rálva ezeket a meglévő rendszerekkel.
* Meglévő webes alkalmazások módosítása, fejlesztése és optimalizálása.
* Rugalmas munkavégzés csapatban és egyénileg egyaránt.
* Kihívást jelentő IT projektek előkészítése, tervezése és kivitelezése.
* Szoftverfejlesztési és tesztelési feladatok magas színvonalon.

### Amit elvárunk:

* Kiváló ismeretek a PHP (verzió = 8.0) nyelven.
* Mélyreható tudás SQL adatbáziskezelés terén.
* Erős alapok HTML, javascript (ES5), CSS terén.
* Magabiztos szakmai angol nyelvtudás.