# Fejlesztői Környezet Dokumentáció

### Címoldal

* **Projekt neve:** FITFORM
* **Készítette:** Iroczki Béla, Vályi Roland Gábor, Radácsi Dávid
* **Dátum:** 2024.11.07

## Tartalomjegyzék

1. Bevezetés
2. Verziókezelés GitHubbal
   1. Git és GitHub alapok
   2. Repository létrehozása és kezelése
   3. Branch kezelés és verziókövetés
   4. Pull Requestek és Review folyamat
   5. Konfliktuskezelés
3. Projektmenedzsment Trellóval
   1. Trello alapok
   2. Feladatkezelés és áramlás
   3. Kanban alapú projektmenedzsment
4. Fejlesztési környezet: Visual Studio Code
   1. Alapvető beállítások
   2. Bővítmények és kiterjesztések
   3. Debugging és hibakeresés
   4. Kódformázás és linting
5. Adatbázis-kezelés MySQL-ben XAMPP segítségével
   1. XAMPP telepítése és konfigurálása
   2. MySQL és PhpMyAdmin használata
   3. SQL lekérdezések és műveletek
6. Tesztelés
   1. Tesztelés alapjai
   2. Manuális tesztelés
   3. Automatizált tesztelés és JUnit
7. Dokumentáció
   1. Dokumentáció struktúra
   2. Verziókezelés és nyomon követés
   3. Dokumentáció frissítése
8. Összegzés és jövőbeli fejlesztési lehetőségek
9. Irodalomjegyzék

## 1. Bevezetés

A modern szoftverfejlesztés során egy jól megtervezett és szervezett fejlesztési környezet alapvető fontosságú a projekt sikeres végrehajtásához. A megfelelő eszközök és környezetek segítenek a kód nyomon követésében, a csapat közötti hatékony kommunikációban és a fejlesztési folyamatok optimalizálásában. Ezen dokumentáció célja a projektben használt eszközök és technológiák bemutatása, beleértve azok előnyeit, alkalmazását és integrációját.

A projekt során a verziókezeléshez a **GitHub**-ot, projektmenedzsmenthez a **Trello**-t, a fejlesztési környezethez a **Visual Studio Code**-ot, valamint adatbázis-kezeléshez az **XAMPP** csomagot és a **MySQL**-t használjuk. Emellett a dokumentáció eszközként a **Microsoft Word**-ot alkalmazzuk. A célunk, hogy az eszközök szoros integrációja és megfelelő alkalmazása révén a projekt zökkenőmentesen és hatékonyan haladjon előre.

## 2. Verziókezelés GitHubbal

### 2.1. Git és GitHub alapok

A verziókezelés egy olyan technológiai megoldás, amely lehetővé teszi a fejlesztők számára a kód történetének követését, módosítását és visszaállítását. A **Git** egy elosztott verziókezelő rendszer, amely lehetővé teszi a fejlesztők számára, hogy párhuzamosan dolgozzanak egy kódon anélkül, hogy zavarnák egymást. A **GitHub** pedig egy távoli verziókezelő szolgáltatás, amely webes felületet biztosít a Git használatához, segítve a csapatok közötti együttműködést és a kód könnyű nyomon követését.

* **Git:** Egy helyi verziókezelő rendszer, amely minden fejlesztő számára biztosít egy saját példányt a kódból, és lehetőséget ad arra, hogy egyéni módosításokat végezzenek, mielőtt azokat a központi tárolóba feltöltik.
* **GitHub:** A Git alapú tároló rendszer, amely lehetővé teszi a központi repository-ban tárolt kód könnyű megosztását és együttműködést.

### 2.2. Repository létrehozása és kezelése

A verziókezelés egyik alapvető lépése a **repository** (tároló) létrehozása. A GitHub lehetőséget ad arra, hogy egy új repository-t hozzunk létre, amely tartalmazza a projekt összes fájlját és annak történelmét. A következő lépések szükségesek a repository kezeléséhez:

1. **git init** – Hozz létre egy új repository-t helyben.
2. **git clone <repository\_url>** – Klónozd le a távoli repository-t helyben.
3. **git add .** – Hozzáadja az összes fájlt a commit-hoz.
4. **git commit -m "commit üzenet"** – Mentse el a módosításokat.
5. **git push origin main** – Töltsd fel a helyi módosításokat a GitHub-ra.

### 2.3. Branch kezelés és verziókövetés

A **branch** (ág) kezelés az egyik alapvető jellemzője a Git-nek, amely lehetővé teszi a párhuzamos fejlesztést. Az ágak lehetővé teszik, hogy egy adott funkciót külön, elkülönítve fejlesszünk, majd később összeolvasszuk a fő ággal (main branch).

* **git branch <branch\_name>** – Új ág létrehozása.
* **git checkout <branch\_name>** – Átváltás egy másik ágra.
* **git merge <branch\_name>** – Két ág egyesítése.

### 2.4. Pull Requestek és Review folyamat

A **Pull Request** egy olyan folyamat, ahol a fejlesztők összesítik az ágak változtatásait, és kérik a kód átnézését és elfogadását a csapattagok részéről. Ez segít biztosítani a kód minőségét és hibamentességét.

### 2.5. Konfliktuskezelés

A verziókezelés során előfordulhatnak konfliktusok, ha két fejlesztő ugyanazon kódrészen dolgozik. A Git lehetőséget ad a konfliktusok kézi feloldására, amely során a fejlesztők eldönthetik, hogy melyik változtatást fogadják el.

## 3. Projektmenedzsment Trellóval

### 3.1. Trello alapok

A **Trello** egy vizuális projektmenedzsment eszköz, amely lehetővé teszi a feladatok könnyed kezelését és követését. A Trello alapját a **Kanban módszertan** képezi, amely segít a feladatok áramlásának optimalizálásában.

A Trello alapvető egységei a **Boards**, **Lists**, és **Cards**, amelyek segítenek rendszerezni a feladatokat.

### 3.2. Feladatkezelés és áramlás

A Trello egyik alapvető funkciója, hogy lehetővé teszi a feladatok kezelését és áramlását a különböző listák között. Minden egyes kártya tartalmazza a feladat részletes leírását, felelős személyét és határidejét.

### 3.3. Kanban alapú projektmenedzsment

A **Kanban** módszertan segíti a csapatokat abban, hogy folyamatosan figyeljék a feladatok előrehaladását, és gyorsan reagáljanak a változásokra. A Trello segítségével könnyedén nyomon követhetjük a projekt állapotát, és biztosíthatjuk a hatékony munkavégzést

## 4. Fejlesztési környezet: Visual Studio Code

### 4.1. Alapvető beállítások

A **Visual Studio Code (VSCode)** egy könnyen használható fejlesztői környezet, amely támogatja a különböző programozási nyelveket és keretrendszereket. A VSCode sokféle beállítást és bővítményt kínál a fejlesztők számára, amelyek segítenek a munka hatékonyságának növelésében.

### 4.2. Bővítmények és kiterjesztések

A VSCode egyik legnagyobb előnye a bővítmények gazdagsága. A következő bővítmények segítik a fejlesztést:

* **ESLint**: A kódformázás és linting segít, hogy a kód megfeleljen a szabályoknak.
* **Prettier**: Egy másik kódformázó bővítmény, amely segít a kód stílusának egységesítésében.
* **Live Server**: A fejlesztés közben automatikusan frissíti a böngészőt, így valós időben láthatjuk a változásokat.

### 4.3. Debugging és hibakeresés

A VSCode hibakereső funkciója segít a fejlesztőknek abban, hogy gyorsan megtalálják a hibákat és problémákat a kódban. Az eszközök közé tartozik a **breakpoint**, amely lehetővé teszi, hogy megállítsuk a kódot egy adott ponton, és nyomon kövessük annak működését.

### 4.4. Kódformázás és linting

A kódformázás és linting eszközök segítenek abban, hogy a kód tiszta és olvasható legyen. A VSCode és a bővítmények biztosítják, hogy minden kód megfeleljen a meghatározott szabványoknak.

## 5. Adatbázis-kezelés MySQL-ben XAMPP segítségével

### 5.1. XAMPP telepítése és konfigurálása

Az **XAMPP** egy népszerű ingyenes szervercsomag, amely tartalmazza az Apache HTTP Server-t, MySQL adatbázist és PHP-t. A telepítés után a phpMyAdmin segítségével könnyedén kezelhetjük az adatbázisokat.

### 5.2. MySQL és PhpMyAdmin használata

A **MySQL** egy relációs adatbázis-kezelő rendszer, amely biztosítja az adatok tárolását és lekérdezését. A **phpMyAdmin** egy webalapú eszköz, amely lehetővé teszi az adatbázisok és táblák kezelését.

### 5.3. SQL lekérdezések és műveletek

Az SQL lehetővé teszi az adatok keresését, módosítását és törlését. A következő alapvető SQL parancsokat használhatjuk:

* **SELECT**: Adatok lekérése.
* **INSERT**: Adatok hozzáadása.
* **UPDATE**: Adatok módosítása.
* **DELETE**: Adatok törlése.

## 6. Tesztelés

### 6.1. Tesztelés alapjai

A tesztelés célja a program működésének biztosítása, hibák felderítése és a szoftver minőségének javítása. Két fő típusú tesztelést végezhetünk: **manuális tesztelés** és **automatizált tesztelés**.

### 6.2. Manuális tesztelés

A manuális tesztelés során a fejlesztők kézzel végzik el a tesztelést, hogy ellenőrizzék a rendszer helyes működését. A manuális tesztelés egyszerűbb alkalmazásokat, kisebb projekteket érinthet.

### 6.3. Automatizált tesztelés és JUnit

Az **automatizált tesztelés** során előre megírt teszt szkriptek segítségével ellenőrizhetjük a program működését. A **JUnit** egy elterjedt tesztelési keretrendszer Java alkalmazásokhoz, amely segíti a tesztelési folyamat automatizálását.

## 7. Dokumentáció

### 7.1. Dokumentáció struktúra

A dokumentáció célja, hogy minden fontos információt rögzítsen a projektről és annak környezetéről. A dokumentáció alapvető részei a kód leírása, a rendszer működése és a telepítési útmutató.

### 7.2. Verziókezelés és nyomon követés

A verziókezelés a dokumentációban is fontos szerepet kap, mivel minden módosítás, kiegészítés és frissítés nyomon követhető.

### 7.3. Dokumentáció frissítése

A dokumentációt rendszeresen frissíteni kell, hogy mindig tükrözze az aktuális állapotot és a fejlesztés legújabb eredményeit.

## 8. Összegzés és jövőbeli fejlesztési lehetőségek

A fejlesztési környezet és eszközök megválasztása alapvető hatással van a projekt sikerére. A GitHub, Trello, Visual Studio Code, MySQL és XAMPP együttes alkalmazása segíti a projekt gördülékeny lebonyolítását. A jövőben a tesztelés automatizálása és a folyamatos integráció bevezetése is lehetőségként merülhet fel.

## 9. Irodalomjegyzék

1. **Visual Studio Code Documentation:** <https://code.visualstudio.com/docs>
2. **XAMPP Official Site:** <https://www.apachefriends.org>
3. **MySQL Documentation:** <https://dev.mysql.com/doc>
4. **GitHub Guides:**  <https://guides.github.com>
5. **Trello Help:**  https://trello.com/guide