Vizsgaszervező:

**Mátészalkai SZC Gépészeti Technikum és Kollégium**

Képző:

**Mátészalkai SZC Gépészeti Technikum és Kollégium**

|  |
| --- |
| Szakképesítés / Szakma megnevezése és azonosítószáma:  ***Szoftverfejlesztő- és tesztelő technikus***  ***5 0613 12 03*** |

**Projektfeladat (gyakorlat)**

|  |
| --- |
| Vizsgafeladat megnevezése:  **Szoftverfejlesztő és -tesztelő projektfeladat** |

**2024. év**

A projektfeladat (gyakorlat) vizsgatevékenység időtartama: **30** perc

Készítette: Ellenőrizte:

Oktató Szakmai vizsgáért felelős vezető

Jóváhagyta:

Igazgató

p.h.

Tartalom

Bevezetés……………………………………………………………………………………………………..3

Futási környezet……………………………………………………………………………………………...4

Fejlesztői környezet………………………………………………………………………………………….7

Technikai referenciák………………………………………………………………………………………11

Adatbázis struktúra…………………………………………………………………………………………12

Forráskód……………………………………………………………………………………………………13

Frissítési és karbantartási útmutató……………………………………………………………………….16

**Fejlesztői dokumentáció**

**Bevezetés**

Üdvözöljük a **Lethal Omen** nevezetű weboldalunk fejlesztői dokumentációjában! Ez a dokumentáció részletes útmutatást nyújt az oldalon található összes funkció használatához és az elérhető feladatok végrehajtásához. A dokumentáció segítséget nyújt a fejlesztőknek a weboldal megértésében és karbantartásában, valamint az új funkciók és fejlesztések hozzáadásában.

A Lethal Omen egy olyan weboldal, amely lehetővé teszi a felhasználók számára, hogy a kártyáinkat megvegyék online, trade-eljenek velük.

**Futási környezet**

Ez a fejezet bemutatja a weboldal futási környezetét, azaz azt a környezetet, ahol a weboldal működik és üzemel. A futási környezet magába foglalja azokat az eszközöket, szoftvereket, és szolgáltatásokat, amelyekre a weboldalnak szüksége van a működése során.

**Hardveres követelmények**

A weboldal működéséhez minimális hardveres követelmények szükségesek.

Az alábbiakban felsoroljuk ezeket:

* ***Szerver:*** A weboldal futtatásához szükséges egy olyan számítógép, laptop, telefon, tablet vagy szerver, amely képes a weboldal folyamatos futtatására.
* ***Memória:*** Ajánlott legalább 2 GB RAM az optimális futtatás eléréséhez.
* ***Tárhely:*** Ajánlott legalább 1 GB szabad tárhely a weblap fájljainak és adatbázisának tárolásához.

**Szoftveres követelmények**

A weboldal futtatásához szükséges szoftveres követelmények a következők:

* ***Operációs rendszer:*** A weboldal kompatibilis az alábbi operációs rendszerekkel: Windows, Linux, macOS.
* ***Webszerver:*** Szükséges egy kompatibilis webszerver a HTTP kérések fogadásához és a weboldalak kiszolgálásához. Például: Apache.
* ***Adatbázis szerver:*** A weboldal támogatja az alábbi adatbázis szervert: MySQL.
* ***Programozási nyelv és keretrendszer:*** A weboldal PHP nyelven íródott és a Laravel keretrendszert használja.

**Támogatott böngészők**

* Google Chrome
* Mozilla Firefox
* Microsoft Edge
* Apple Safari
* Opera, Opera GX

**Telepítés és konfiguráció**

Ez a szakasz részletesen ismerteti a weboldal telepítési és konfigurációs folyamatát. A weboldal helyes telepítése és beállítása elengedhetetlen a megfelelő működéshez.

1. ***Szükséges szoftverek előkészítése:***

-Ellenőrizze, hogy a szerver megfelel-e a hardveres és szoftveres követelményeknek.

-Telepítse és konfigurálja a szükséges operációs rendszert, támogatott böngészőt/böngészőket, webszervereket és adatbázis szerkezeteket.

1. ***Weboldal letöltése:***

-Töltse le a weboldal forráskódját a megfelelő forrásból. Például GitHub repository.

-Másolja/telepítse a forráskódot a szerver kívánt mappájába.

1. ***Környezet konfigurációja:***

-Állítsa be a szükséges környezeti változókat és beállításokat a weboldal futtatásához.

-Konfigurálja az adatbázis kapcsolódási paramétereit, URL-jét, felhasználónevét és jelszavát.

1. ***Függőségek telepítése:***

-Telepítse a weboldal függőségeit, például PHP csomagokat vagy JavaScript könyvtárakat.

-Győződjön meg róla, hogy minden szükséges függőség telepítve van a rendszerben.

1. ***Adatbázis inicializálása:***

-Hozza létre az alapvető adatbázist és az alapvető táblákat, ha még nem tette meg.

-Futtassa az adatbázis inicializáló parancsokat vagy scripteket a szükséges adatbázis struktúra létrehozásához.

1. ***Tesztelés és finomhangolás:***

-Tesztelje a telepített weboldalt, hogy megbizonyosodjon a helyes működésről és konfigurációról.

-Végezze el a szükséges finomhangolásokat és beállításokat az optimális működés érdekében.

**Hibakeresés és monitorozás**

Fontos biztosítani, hogy a weboldal futási környezete stabil legyen, és az esetleges hibák gyorsan javíthatók legyenek. Ezért javasolt hibakeresési és monitorozási eszközök beállítása az eszközön, rendszeren.

**Biztonság**

A weboldal futási környezetének biztonsága nagyon fontos. Szükséges megfelelő biztonsági intézkedéseket beállítani, például tűzfalak, SSL tanúsítványok, és megfelelő jogosultságkezelés a szerveren és az oldalon.

**Egyéb fontos információk**

**JavaScript:** A weboldalunk JavaScript-alapú funkciókat használ, és támogatja a böngészőkben engedélyezett JavaScript futtatást.

**Fejlesztői környezet**

A fejlesztői környezet részletes leírása azoknak a lépéseknek és eszközöknek az összessége, amelyekre szükség van ahhoz, hogy sikeresen beállítsuk és konfiguráljuk a fejlesztői környezetet a projektben való hatékony munkához.

1. ***Fejlesztői eszközök:***

* A projekt fejlesztéséhez **szükséges eszközök**: Visual Studio Code
* IDE-k (**Integrated Development Environmen**t): Visual Studio Code
* **Verzió kezelő rendszerek**: Git
* **Kódtároló platform**: GitHub
* **Tesztelési eszköz**: Bármely támogatott böngésző, például Google Chrome
* **Hibakereső eszközök**: Visual Studio Debugger
* **Projektmenedzsment:** GitHub, Discord

1. ***Környezet beállítása:***

* **Szükséges szoftverek telepítése:**

1. **Node.js és npm**: Először is látogasson el a Node.js hivatalos weboldalára(<https://nodejs.org/>), és telepítse a legfrissebb stabil verziót. Ez telepíteni fogja a Node.js-t és az npm-et is.
2. **Git**: Telepítse a Git verziókezelő rendszert a Git hivatalos weboldaláról(<https://git-scm.com/>). Ez lehetővé teszi a kódbázis verzióinak kezelését.
3. **Projekt klónozása**: Nyissa meg a parancssor alkalmazást, és navigáljon a mappába, ahol a projektet szeretné tárolni.   
     
   -Klónozza a projektet a GitHub-ról a következő parancssorral:   
     
    **git clone <repository\_URL>**
4. **Függőségek telepítése:** Navigáljon a projekt gyökérkönyvtárába a parancssorban.  
   Telepítse a projekt függőségeit az alábbi parancs segítségével:  
     
   **npm install**
5. **Konfiguráció:**   
   **Node.js és npm beállítása:**  
   - Ellenőrizze a **Node.js** és **npm** telepítését a parancssorban a következő parancssal:  
     
   **node -v  
   npm -v**  
    **Git beállítása:**- Állítsa be a Git felhasználói nevét és e-mail címét a következő parancssal:  
     
   **git config –global user.name „Your Name”  
     
   git config –global user.email „**[**your\_email@example.com**](mailto:your_email@example.com)**”**
6. **Projekt környezet indítása:** Ha van lehetőség a helyi fejlesztői szerver indítására, indítsa el a projektet a következő paranccsal:  
     
   **npm start**
7. **Tesztelés:  
   -**Nyissa mega weboldalt, és ellenőrizze, hogy a projekt megfelelően működik-e a helyi fejlesztői környezetben.
8. ***Függőségek telepítése:*** 
   1. **Parancssor megnyitása:** Nyissa meg a parancssor alkalmazást.
   2. **Navigálás a projekt gyökérkönyvtárába:** Használja a parancssort a projekt gyökérkönyvtárába való navigálásához. Írja be ezt a parancssort:  
        
      **cd C:\Users\Felhasznaloneve\Dokumentumok\lethalomen**
   3. **Függőségek telepítése npm segítségével:   
        
      npm install**Ez a parancs az „**npm install**” parancs, amely a **package.json** fájlban meghatározott összes függőséget telepíti. A **package.json** fájl a projekt gyökérkönyvtárában található, és tartalmazza a projekt metaadatait és függőségeit.  
      Ha a projekt függőségei már listázva vannak a **package.json** fájlban, akkor a fent említett parancs kiadása után az **npm** letölti és telepíti az összes szükséges függőséget a „**node\_modules”** mappába a projekt gyökérkönyvtárába.
   4. **Ellenőrzés:** Ellenőrizze, hogy a függőségek sikeresen települtek-e a projektbe. Navigáljon a projekt gyökérkönyvtárába, és keresse meg a „**node\_modules”** mappát. Ha ez a mappa létezik, és tartalmazza a projekt függőségeit, akkor a telepítés sikeres volt.  
        
      Ezzel a folyamattal a projekt összes függősége telepítve lesz a helyi gépre, és a fejlesztők készen állnak a projekt fejlesztésére és tesztelésére a helyi környezetben.
9. ***Adatbázis konfiguráció:*** 
   1. **Adatbázis szoftver telepítése:**  Látogasson el a **MySQL** hivatalos weboldalára, és töltse le/ telepítse le.
   2. **Adatbázis szolgáltatás indítása:** Miután telepítette az adatbázis szoftvert, indítsa el azt.   
        
      Futtassa a **MySQL** szolgáltatást a következő paranccsal:   
        
      **mysql.server start**
   3. **Adatbázis konfiguráció:**  Miután az adatbázis szolgáltatás fut, konfigurálja az adatbázist a projekt számára.  
        
      Nyisson meg egy parancssort, és jelentkezzen be az adatbázis szerverére az adminisztrátori jogosultságokkal. Futtassa a következő parancsot:   
        
      **mysql – root -p**Majd adja meg jelszavát.  
        
      Hozzon létre új adatbázist a projekt számára.   
        
      **CREATE DATABASE exampledb;**Frissítse a **.env** fájlt a projektben, hogy az alkalmazás tudja, hogy hogyan kapcsolódjon az adatbázishoz. Adja meg az adatbázis nevét, felhasználónevét és jelszavát.
   4. **Tesztelés:** Ellenőrizze, hogy az adatbázis konfigurációja sikeres volt-e, és hogy az alkalmazás képes-e kommunikálni az adatbázissal.   
        
      Futtassa az alkalmazást, és olvasson adatokat az adatbázisban.  
        
      Ellenőrizze az adatbázisban található adatokat, és győződjön meg róla, hogy helyesen működnek-e.  
        
      Ezzel az adatbázis konfigurációja befejeződött, és az alkalmazás készen áll a fejlesztésre a helyi környezetben az adatbázissal való kommunikációhoz.
10. ***Tesztelési környezet:***  
    **a.**. **Tesztelési keretrendszer kiválasztása:** Válasszon ki egy megfelelő tesztelési keretrendszert a projekt számára. Ez lehet manuális vagy automatizált tesztelési keretrendszer, vagy akár mindkettő.   
      
    Ha automatizált tesztelési keretrendszert szeretne használni, akkor itt van néhány elterjedt tesztelési keretrendszer: **PHPUnit, Jest, Mocha.**  
     **b. Tesztelési környezet beállítása:** Telepítse és konfigurálja a kiválasztott tesztelési keretrendszert a projekt számára. Kövesse a tesztelési keretrendszer dokumentációját az előírt beállításokhoz és konfigurációjához.  
      
    **c. Tesztesetek létrehozása:** Hozzon létre teszteseteket a projekt különböző részeihez és funkcióihoz. A tesztesetek legyenek jól meghatározottak, és fedjék fel az összes lehetséges forgatókönyvet és kimenetet.  
      
    Ha automatizált tesztelési keretrendszert használ, implementálja a teszteseteket a keretrendszer segítségével.

**d. Tesztadatok előkészítése:** Készítsen elő teszt adatokat a teszteléshez. Ezek lehetnek például adatbázisbejegyzések vagy más források, amelyekre a tesztek támaszkodhatnak.  
  
**e. Tesztelés futtatása:** Futtassa le a tesztelési keretrendszerrel a teszteseteket.  
Győződjön meg róla, hogy a tesztek sikeresen lefutottak-e, és az összes teszteset a projekt követelményeinek megfelelően működik-e.   
  
**f. Hibajavítás és iteráció:** Ha a tesztek során hibák vagy hiányosságok merülnek fel, javítsa ki ezeket és futtassa újra a teszteket.   
Ismételje meg ezt az iterációt, amíg minden teszteset sikeresen nem teljesül és a projekt minősége elfogadható szinten van.

**Technikai referenciák**

1. ***Felhasznált technológiák:*Laravel 8.2.12** használata **PHP 7.3** környezetben, **Composer** a függőségek kezeléséhez.
2. ***Routing és Controller:*** Példa:  
     
   **Route::get(‘/omen’, function() {  
    return view(‘omen’);  
   });**
3. ***Adatbázis modell és migrációk:*** Példa:   
     
   **Schema::create('password\_reset\_tokens', function (Blueprint $table) {**

**$table->string('email')->primary();**

**$table->string('token');**

**$table->timestamp('created\_at')->nullable();  
});**

1. ***Telepítési útmutató:***  Laravel telepítése:   
     
   **composer create-project laravel/laravel “projectneve”**
2. ***Tesztelés és hibakeresés:***  **PHPUnit** tesztek futtatása:   
     
   **php artisan test**
3. ***Verziókezelés:*** A kód verziókezelése a **GitHub**-on.

**Adatbázis struktúra**

1. ***Felhasználók(users) tábla***  
     
   **CREATE TABLE users(  
    id INT PRIMARY KEY AUTO\_INCREMENT,  
    nev VARCHAR(50) ,**

**email VARCHAR(100) NOT NULL,  
 jelszo VARCHAR(255) NOT NULL,  
 gyujtemeny INT(10) ,  
)**

1. ***Kártyák(kartyak)*CREATE TABLE kartyak(  
    id INT PRIMARY KEY AUTO\_INCREMENT,  
    kartyanev VARCHAR(100) NOT NULL,  
    ar INT(1000000) NOT NULL,  
    klan VARCHAR(10) NOT NULL,  
     
   )**
2. ***Kereskedés(kereskedes)*CREATE TABLE kereskedes(  
    id INT PRIMARY KEY AUTO\_INCREMENT,  
    idopont DATETIME NOT NULL,  
    kereskedok INT(100) NOT NULL,  
    mitmire INT(10) NOT NULL,  
     
     
   )**

**Forráskód**

1. ***Felhasználó model***

**<?php**

**/\*Adat validálás\*/**

**namespace App\Models;**

**// use Illuminate\Contracts\Auth\MustVerifyEmail;**

**use Illuminate\Database\Eloquent\Factories\HasFactory;**

**use Illuminate\Foundation\Auth\User as Authenticatable;**

**use Illuminate\Notifications\Notifiable;**

**class User extends Authenticatable**

**{**

**use HasFactory, Notifiable;**

**/\*\***

**\* A tömegesen hozzárendelhető attribútumok.**

**\***

**\* @var array<int, string>**

**\*/**

**protected $fillable = [**

**'name',**

**'email',**

**'password',**

**];**

**/\*\***

**\* Azok az attribútumok, amelyeket el kell rejteni a szerializáláshoz.**

**\***

**\* @var array<int, string>**

**\*/**

**protected $hidden = [**

**'password',**

**'remember\_token',**

**];**

**/\*\***

**\* Megszerzi az átadandó attribútumokat.**

**\***

**\* @return array<string, string>**

**\*/**

**protected function casts(): array**

**{**

**return [**

**'email\_verified\_at' => 'datetime',**

**'password' => 'hashed',**

**];**

**}**

**}**

1. ***“Home” kontroller*<?php**

**namespace App\Http\Controllers;**

**use Illuminate\Http\Request;**

**use Illuminate\Foundation\Bootstrap;**

**class HomeController extends Controller**

**{**

**/\*\***

**\* Create a new controller instance.**

**\***

**\* @return void**

**\*/**

**public function \_\_construct()**

**{**

**$this->middleware('auth');**

**}**

**/\*\***

**\* Show the application dashboard.**

**\***

**\* @return \Illuminate\Contracts\Support\Renderable**

**\*/**

**public function index()**

**{ return view('home');}}**

1. ***Weboldal nézet***

**@extends('layouts.app')**

**@section('content')**

**<div class="container mt-5 p-4">**

**<div class="row justify-content-center">**

**<div class="col-md-6">**

**<div class="input-group mb-3">**

**<input type="search" class="form-control" placeholder="Keresés..." aria-label="Keresés..." aria-describedby="button-addon2">**

**<button class="btn btn-outline-secondary" type="button" id="button-addon2">Keresés</button>**

**</div>**

**</div>**

**</div>**

**</div>**

**@foreach ($adat as $kartya)**

**<div class="cards">**

**<p class="cards2">{{$kartya->Kártyanév}}</p>**

**<img src="{{ asset('storage/kartyak/'.$kartya->Kártyakép) }}">**

**</div>**

**@endforeach**

**@endsection**

**Frissítési és karbantartási** **útmutató**

**1. Rendszerkövetelmények ellenőrzése:**

Mindig győződj meg arról, hogy az alkalmazásod fut a legfrissebb, támogatott környezetben. Ellenőrizd a követelményeket, például a PHP verziót, a Laravel verziót, a szükséges kiegészítőket és a szerver konfigurációját.

**2. Biztonsági Frissítések:**

Kövesd figyelemmel a Laravel és az alkalmazásod által használt függőségek biztonsági frissítéseit. A Laravel rendszeresen kiad biztonsági frissítéseket, amelyeket fontos azonnal telepíteni a rendszer biztonságának fenntartása érdekében.

**3. Laravel frissítése:**

Frissítsd a Laravel keretrendszert a legújabb verzióra a Laravel hivatalos dokumentáció szerint. Győződj meg róla, hogy a saját alkalmazásodhoz és egyéb függőségeidhez igazítsd a kódodat és a konfigurációkat az új verzióhoz.

**Példa:**

**composer update laravel/framework --with-dependencies**

**4. Adatbázis migrációk:**

Futtasd az adatbázis migrációkat az esetleges adatbázis-séma változások alkalmazásához vagy frissítéséhez.

**php artisan migrate**

**5. Ellenőrzés és tesztelés:**

Teszteld az alkalmazást, hogy megbizonyosodj róla, hogy az új verzió működik az elvárt módon, és nincsenek hibák vagy problémák.

**6. Teljesítmény optimalizálás:**

Vizsgáld meg az alkalmazás teljesítményét és optimalizáld szükség esetén. Használhatsz Laravel beépített eszközöket, például a cache-eket és a lapokat, vagy kiegészítő eszközöket, például a Laravel Debugbar-t.

**7. Biztonsági másolatok készítése:**

Készíts rendszeres biztonsági másolatokat az alkalmazásodról és az adatbázisról, hogy visszatérhess a korábbi állapotba, ha valami hiba történik.

**8. Hibák javítása:**

Ha a frissítés után bármilyen hiba vagy probléma jelentkezik, javítsd ki azokat a lehető leghamarabb. Használd a naplófájlokat és a hibakeresési eszközöket a hibák azonosításához és javításához.

**9. Dokumentáció Frissítése:**

Ne felejtsd el frissíteni a dokumentációt az alkalmazás változásai szerint, hogy az mindig naprakész legyen a fejlesztők és az üzemeltetők számára.