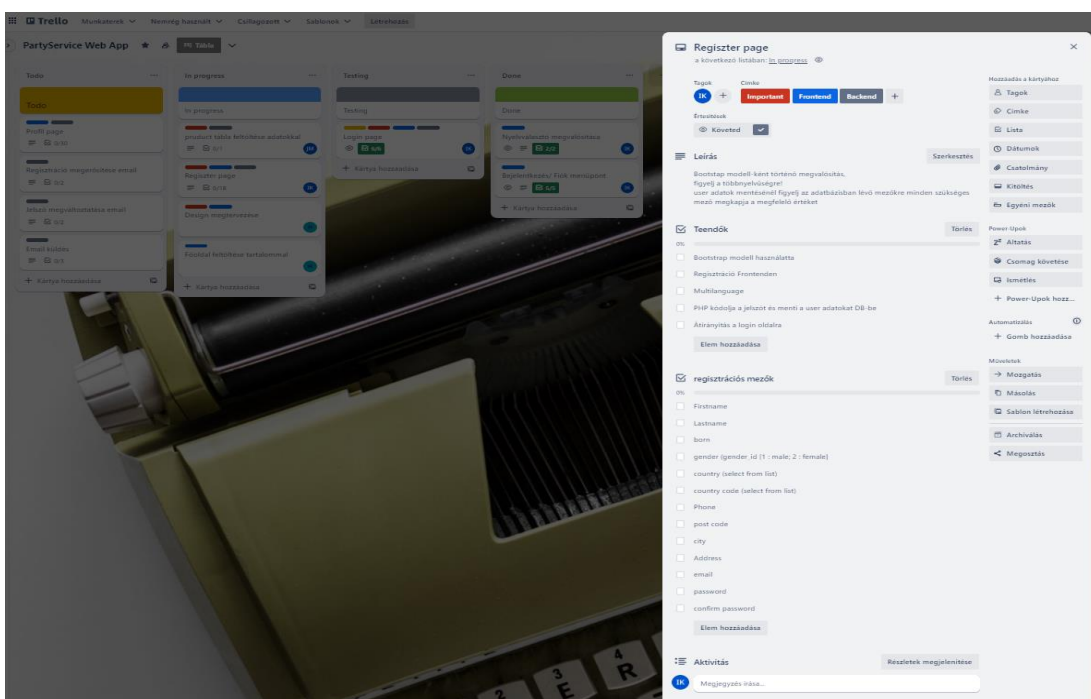
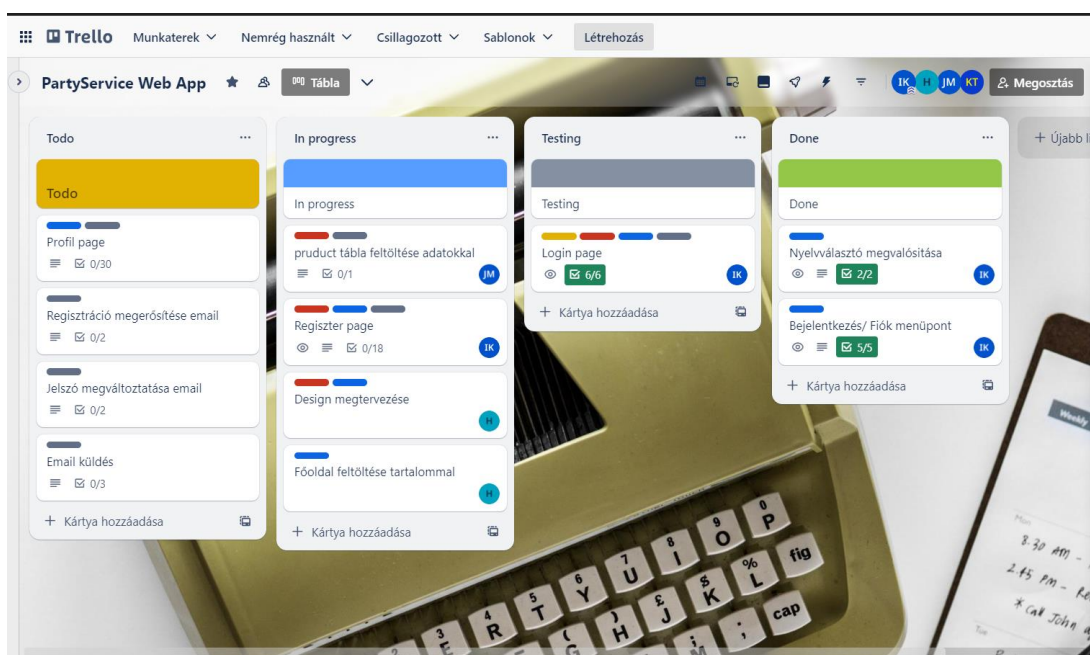


# Technikai dokumentáció a Party Service oldalhoz

## Közös munka és átláthatóság:

közös munka gördülékenysége érdekében és a feladatok jobb átláthatósága miatt feladatok kiosztását a Trello segítségével oldottuk meg, verziókezeléshez a Git-tet használtuk.



# 1. Bevezetés

## 1.1 Cél

Ez a technikai dokumentáció a weboldal fejlesztői és karbantartói számára készült, hogy megértse a rendszer működését és karbantarthatóságát.

## 1.2 Közönség

Ez a dokumentáció a fejlesztők, rendszermérnökök és karbantartók számára készült.

# 2. Architektúra és Technológiák

## 2.1 Rendszerarchitektúra

### 2.1.1. Felhasználói Interfész (UI)

A felhasználói interfész réteg felelős a weboldal vizuális megjelenítéséért és az interakcióért a felhasználóval. A weboldalunk responzív, és a következő technológiákat alkalmazza:

- **Frontend Framework:** A felhasználói felületet Angular JS keretrendszerrel valósítjuk meg.
- **CSS Keretrendszer:** A stílusok és elrendezések kezelésére Bootstrap keretrendszert alkalmazunk.
- **Interaktív Elemek:** Az oldal interaktivitását JavaScript segítségével biztosítjuk, és AJAX-t használunk az aszinkron adatátvitelhez.

### 2.1.2. Alkalmazás Logika

Az alkalmazás logika réteg tartalmazza a frontend és backend alkalmazásokat. Az oldal elkészítése során törekedtünk a tiszta kód elvének betartására.

- **Frontend Réteg:** A frontend alkalmazás a felhasználói interfészéért felelős, valamint az felhasználói oldalon történő ellenőrzéseket és adatok feldolgozását végzi.
- **Backend Réteg:** A backend réteg felelős a felhasználói kérések feldolgozásáért, az adatbázishoz való hozzáférésért és az üzleti logika végrehajtásáért. A backendet PHP környezetben valósítjuk meg.

### 2.1.3. Adatbázis

Az adatbázis réteg tárolja és kezeli az alkalmazás által használt adatokat.

- **Típus és Struktúra:** Az alkalmazás adatbázisához MySQL relációs adatbázisrendszert használunk. Az adatok struktúráját szigorúan definiált séma szerint kezeljük.
- **ORM (Object-Relational Mapping):** Az adatbázis és az alkalmazás közötti kommunikációhoz PHP-t alkalmazunk.

## 2.2 Technológiák

### Frontend:

- HTML 5
- CSS 3
- Bootstrap 5.3.2
- JavaScript
- Angular Js 1.8.2
- Angular ui Router 1.0.3
- Jquery 3.7.1
- jquery-ui 1.13.2
- moment Js 2.29.4

### Backend:

- PHP

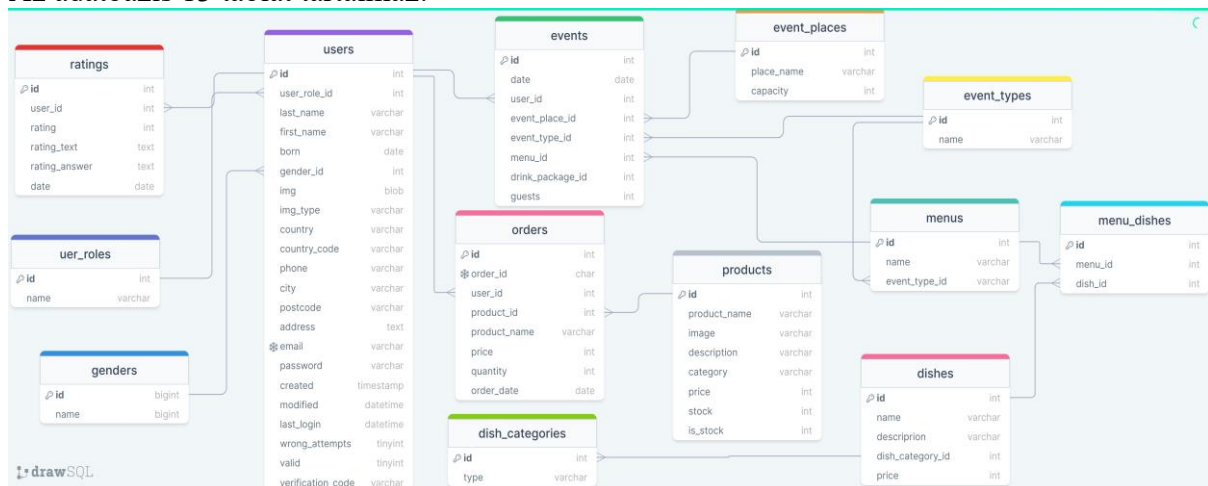
### Adatbázis

- MySQL

## 3. Adatbázis

### 3.1 Adatbázis

Az adatbázis 13 táblát tartalmaz:



az adatbázisunk tartalmaz még egy tárolt eljárást melyet a bejelentkezésnél használunk a profilkép lekéréséhez

```
1 BEGIN
2 /*
3  Convert blob to base64 text, remove start, end spaces,
4  newline, carriage return, and tab characters from text
5 */
6 DECLARE textOut LONGTEXT CHARSET utf8mb4 DEFAULT '';
7 IF (textIn IS NOT NULL) THEN
8   SET textOut = TO_BASE64(textIn);
9   SET textOut = TRIM(textOut);
10  IF (LENGTH(textOut) > 0) THEN
11    SET textOut = REPLACE(textOut, "\n", "");
12    SET textOut = REPLACE(textOut, "\r", "");
13    SET textOut = REPLACE(textOut, "\t", "");
14  END IF;
15 END IF;
16 RETURN textOut;
```

## 4. API Dokumentáció

### 4.1 API Leírás

#### 4.1.1 Regisztráció

A regisztrációhoz ki kell töltenünk minden szükséges mezőt majd a regisztrációra kattintani. A mezők ellenőrzését az Angular Js végzi a háttérben:

- required mező esetén csak azt ellenőrizzük hogy legyen benne adat.
- ng-pattern esetén átadunk egy regexet is melynek meg kell felelni

ng-pattern="/^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z]).{6,20}\$/"

- amennyiben egyedi ellenőrzésre is szükség van pl.: jelszó és jelszó megerősítés esetén a mezőknek egyezni kell az Angular Js validátorát ki tudjuk egészíteni:

html: ng-change="validatePasswordConfirm()"

js: készítünk egy egyedi ellenőrzést

```
<input id="reg_password_confirm"
      type="{{inputType}}"
      class="form-control rounded-pill shadow-sm"
      ng-attr-placeholder="{{('password' | translate:lang.data +
      ' ' + 'confirmation' | translate:lang.data)}}"
      spellcheck="false"
      autocomplete="off"
      ng-model="values.passwordConfirm"
      ng-change="validatePasswordConfirm()"
      ng-pattern="/^(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{6,20}$/"
      name="passwordConfirm"
      ng-maxlength="20"
      required>

$scope.validatePasswordConfirm = () => {
  const { password, passwordConfirm } = $scope.values;
  $scope.registerForm.passwordConfirm.$setValidity(
    'passwordMismatch',
    password === passwordConfirm
  );
};
```

#### A regisztrációra kattintva elindítunk egy AJAX kérést

- url: './php/register.php'
- method: 'POST'
- data: \$scope.userData (body)
- response:
  - amennyiben olyan email címet adtunk meg mely már létezik akkor visszatér 'the user already exist' üzenettel
  - sikeres regisztráció esetén a 'registration successful' üzenettel.

A háttérben a php kapcsolódik a users táblánkhoz PDO kapcsolattal. Elsőként ellenőrzi, hogy a megadott email cím létezik-e az adatbázisban, amennyiben már regisztrált email címről van szó úgy válaszként beállítja a 'the user already exist' választ és ezt továbbítja a frontend felé. Ha nem létezik az email cím akkor a jelszót kódolja, előállítja a created mezőt \$args['created'] = date("Y-m-d H:i:s"); majd az adatokat beírja az adatbázisba és responsnak elküldi a frontend felé a 'registration successful' választ

### 4.1.2 Login

Bejelentkezéshez ki kell tölteni az email cím és jelszó mezőt, melyek ellenőrzése az előzőekben leírtak alapján történik majd a bejelentkezés gombra kattintva következő Ajax kérés fut le:

- url: './php/login.php'
- method: 'Get'
- data: data (body)
- response:
  - Amennyiben hibás email címet írtunk be 'user\_is\_not\_exist' melyet a fronted lefordítva megjelenít egy alert ablakban.
  - Hibás jelszó esetén: 'incorrect\_password', ekkor a rendszer a wrong\_attempts rekordban megnöveli a hibás próbálkozások számát, amennyiben ez meghaladja az 5-öt a rendszer automatikusan letiltja a felhasználót
  - Amennyiben a felhasználó le van tiltva: 'the\_user\_is\_disabled'
  - Sikeres bejelentkezés esetén responseként megkapjuk a felhasználó alap adatait és amennyiben volt hibás próbálkozás a hibás próbálkozások számát 0-ra állítjuk.

### 4.1.3 Profil

A profil oldal megnyitásakor három kérés fut le:

1. profil adatok lekérése:

- url: './php/get\_profil.php'
- method: 'Get'
- data: data (body)
- response: A felhasználó összes adata az alapadatokon kívül.

2. lefoglalt időpontok lekérése:

- url: './php/get\_reservation.php'
- method: 'Get'
- data: {id: \$rootScope.user.id} (body)
- response: A felhasználóhoz tartozó foglalások

3. rendelések lekérése:

- url: './php/get\_orders.php'
- method: 'Get'
- data: {id: \$rootScope.user.id} (body)
- response: A felhasználóhoz megrendelések

Ezután lehetőségünk van saját adatunk szerkesztéséhez és profil kép feltöltéséhez, melynek maximális mérete 64 kbyte, a kép tárolása az adatbázisban történik. Az adatok megváltoztatásánál az ellenőrzés folyamata megegyezik a regisztrációnál végzett ellenőrzéssel. A mentés gombra kattintva tudjuk elindítani az adataink frissítését.

- url: './php/profil.php'
- method: 'POST'
- data: \$scope.userData (body)
- response: Válaszként megkapjuk a beillesztett sorok számát.

Backend oldalon a php legenerálja a változtatás dátumát, a képed decodolja amennyiben töltöttünk fel, majd csatlakozik az adatbázisunkhoz ahol egy update kéréssel módosítja adatainkat.

#### 4.1.4 Értékelések megjelenítése

- url: './php/ratings.php'
- method: 'Get'
- response: Az adatbázisban tárolt értékeléseket adja vissza

#### 4.1.5 Értékelések leadása

- url: './php/send\_rating.php'
- method: 'POST'
- data: \$scope.rating\_data, (body)
- response: Beillesztett sorok száma

Értékelés leadása csak bejelentkezés után lehetséges

#### 4.1.6 Alap menük lekérdezése

- url: './php/menus.php'
- method: 'GET'
- response: Adatbázisunkban tárolt alap menük

```
// Set query
$query = "SELECT menus.id           As id,
        menus.name           As menu,
        dish_categories.type As category,
        dishes.name           As name,
        dishes.description    As description,
        dishes.price          As price
FROM dishes
INNER JOIN dish_categories
    ON dishes.dish_category_id=dish_categories.id
INNER JOIN menu_dishes
    ON dishes.id=menu_dishes.dish_id
INNER JOIN menus
    ON menu_dishes.menu_id=menus.id;";

// Execute query with argument
$raw_result = $db->execute($query);
//echo json_encode($raw_result);
$result = structure_menu_items($raw_result);
```

#### 4.1.7 Időpontfoglalás

Elsőként lekérdezzük az elérhető eseményeket melyekre időpontot tudunk foglalni és a lehetséges helyszíneket.

- url: './php/services.php'
- method: 'GET'
- response: Adatbázisunkban tárolt alap menük

Ezután kitöltjük a megfelelő mezőket, A helyszín kiválasztásánál a rendszer automatikusan elindít egy kérést a szerver felé és lekérdi az adott helyszínre már lefoglalt időpontokat.

The screenshot shows a web form titled 'Időpont foglalás'. At the top, there are two buttons: 'menük' and 'időpont foglalás'. The form fields are as follows:

- Név: Kertész István
- Vendégek száma: Vendégek száma
- Esemény: -- válasszon --
- Helyszín: -- válasszon --
- Dátum: 2024-01-19
- Menü: -- válasszon --
- Italcsomag: -- válasszon --
- A 'küldés' button is located at the bottom right of the form.

- url: './php/check\_days.php'
- method: 'GET'
- response: dátumok listája

A frontend részen jQuery-datepicker-rel letiltjuk a már foglalt időpontokat. Ha minden adat kitöltésre került, elküldhetjük foglalásunk.

- url: './php/reservation.php'
- method: 'POST'
- data: \$scope.reservation (body)
- response: Sikeres rendelés esetén megkapjuk a beillesztett sorok számát ekkor a rendszer egy emailt küld nekünk a rendelésről PHPMailer segítségével és egy alert ablakban jelzi a felhasználó fele hogy sikeresen leadta rendelését. Hiba esetén egy sikertelen rendelés üzenettel tájékoztatja a felhasználót.

```
//check reserved day and set to inactive
$scope.checkDays = () => {
  // Http request check available days
  http
    .request({
      url: './php/check_days.php',
      method: 'POST',
      data: { id: $scope.reservationData.event_place.id },
    })
    .then((response) => {
      const disabledDates = response
      ? response.map((date) => date.date)
      : [];
      $('#date').datepicker('destroy');
      $('#date').datepicker({
        changeMonth: true,
        changeYear: true,
        minDate: new Date($scope.reservDate.min),
        maxDate: new Date($scope.reservDate.max),
        firstDay: 1,
        dayNamesMin:
          $rootScope.lang.id === 'hu'
            ? ['V', 'H', 'K', 'Sze', 'Cs', 'P', 'Szo']
            : $rootScope.lang.id === 'en'
            ? ['Su', 'Mo', 'Tu', 'We', 'Th', 'Fr', 'Sa']
            : ['So', 'Mo', 'Di', 'Mi', 'Do', 'Fr', 'Sa'],
        dateFormat: 'yy-mm-dd',
        beforeShowDay: function (date) {
          const dateString = jQuery.datepicker.formatDate(
            'yy-mm-dd',
            date
          );
          return [disabledDates.indexOf(dateString) === -1];
        },
      });
    });
}
```

#### 4.1.8 webshop/products

- url: './php/product.php'
- method: 'GET'
- response: Az adatbázisunkban tárolt termékek listája

#### 4.1.9 Megrendelés rögzítése

- url: './php/menus.php'
- method: 'POST'
- data: {
 

userId: \$rootScope.user.id,  
 email: \$rootScope.user.email,  
 cart: args,  
 shipping: \$scope.shipping,  
 total: \$scope.getTotalPrice() + \$scope.shipping,  
 lang: { id, type },  
 userName: \$rootScope.user.first\_name,

 },
- response: Sikeres megrendelés esetén a servert üzenetet küld a felhasználónak mely 'email\_sent\_succesful' válasszal tér vissza frontend felé ekkor a felhasználónak egy 'sikeres megrendelés' üzenettel jelezzük a megrendelés megvalósulását. Sikertelen rendelés esetén a felhasználó fele jelezzük hogy megrendelése sikertelen.

## **5. Felhasznált képek**

A projekt során felhasznált képeket többségéről az O-dan oldalról szereztük be, ezen kívül az oldalon felhasználtunk a tengr.ai által generált képeket, illetve egy éttermi szakácstól kapott képeket használtunk fel

## **6. Tesztelés**

Az oldal tesztelését manuális teszteléssel oldottuk meg.

## **7. Csapatunk**

Joó Mária Szoftverfejlesztő

Hevesi Szabolcs Szoftverfejlesztő

Kertész István Szoftverfejlesztő - Projektvezető