



győri szakképzési centrum

Jedlik Ányos  
Gépipari és Informatikai  
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

## Záródolgozat feladatkiírás

Tanuló(k) neve<sup>1</sup>: Németh Csaba, Hejner Bálint, Hidasi Zalán  
Képzés: nappali / felnőttoktatás - esti munkarend  
Szak: 5 0613 12 03 Szoftverfejlesztő és tesztelő technikus

### A záródolgozat címe:

**CareCrop**

Konzulens: Nits László  
Beadási határidő: 2024. 04. 15.

Győr, 2022. 04. 15

---

**Módos Gábor**  
igazgató

---

<sup>1</sup> Szakmajegyzékes záródolgozat esetében több szerzője is lehet a dokumentumnak, OKJ-s záródolgozatnál egyetlen személy ad le záródolgozatot.



győri szakképzési centrum

Jedlik Ányos  
Gépipari és Informatikai  
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

## Konzultációs lap<sup>2</sup>

	A konzultáció		Konzulens alá- írása
	ideje	témája	
1.	2024.02.15.	Témaválasztás és specifikáció	
2.	2024.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2024.04.15.	Dokumentáció véglegesítése	

## Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkánk eredménye. Dolgozatunk azon részeit, melyeket más szerzők munkájából vettünk át, egyértelműen megjelöltük.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul vesszük, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár minket és szakmai vizsgát csak új záródolgozat készítése után tehetünk.

Győr, 2024. április 15.

tanuló aláírása

tanuló aláírása

tanuló aláírása

<sup>2</sup> Szakmajegyzékes, csoportos konzultációs lap



győri szakképzési centrum

Jedlik Ányos  
Gépipari és Informatikai  
Technikum és Kollégium



9021 Győr, Szent István út 7.

+36 (96) 529-480

+36 (96) 529-448

OM: 203037/003

jedlik@jedlik.eu

www.jedlik.eu

## Konzultációs lap<sup>3</sup>

	A konzultáció		Konzulens aláírása
	ideje	témája	
1.	2024.02.15.	Témaválasztás és specifikáció	
2.	2024.03.14.	Záródolgozat készültségi fokának értékelése	
3.	2024.04.15.	Dokumentáció véglegesítése	

## Értékelés

A záródolgozat százalékos értékelése: .....

Legalább 51%-ot elérő előzetes értékelés és három igazolt konzultáció esetén a záródolgozat megfelelt.

Győr, 2024. április 15.

értékelő aláírása

## Tulajdonosi nyilatkozat

Ez a dolgozat a saját munkám eredménye. Dolgozatomban azon részeit, melyeket más szerzők munkájából vettem át, egyértelműen megjelöltem.

Ha kiderülne, hogy ez a nyilatkozat valótlan, tudomásul veszem, hogy a szakmai vizsgabizottság a szakmai vizsgáról kizár és szakmai vizsgát csak új záródolgozat készítése után tehetek.

Győr, 2024. április 15.

tanuló aláírása

<sup>3</sup> OKJ-s, egyéni konzultációs lap



Győri SZC Jedlik Ányos Gépipari és Informatikai  
Technikum és Kollégium



***NEKÜNK A TERMÉNY A FONTOS***

**KÉSZÍTETTE:**

*HEJNER BÁLINT*

*NÉMETH CSABA*

*HIDAS ZALÁN*

# Tartalomjegyzék

<b>1. A vizsgaremekről</b>	<b>3</b>
I. Mi nekünk a CareCrop?	3
II. Mi is a probléma?	3
III. Miért CareCrop?	3
IV. Mi várható a jövőben?	3
<b>2. Adatbázis</b>	<b>4</b>
I. Tervezés	4
II. draw.io	4
III. MariaDB/dbForge	4
IV. Táblák	6
a. Termelők	6
b. Telephelyek	6
c. Termények	6
d. Vásárlás	7
e. Vásárlók	7
f. Support	7
<b>3. Backend</b>	<b>8</b>
I. LAMP	8
II. A backend felépítése	9
a. Hitelesítés a MariaDB adatbázishoz	9
b. Végpontok működése	9
III. Hibakezelés	11
IV. Regisztráció	12
V. Bejelentkezés	13
VI. Jogosultságok	14
VII. Swagger	15
VIII. Git commitok	16
<b>4. Frontend</b>	<b>17</b>
I. Tervezés	17
II. Figma	17
III. Fejlesztői környezetek	18
a. JavaScript	18
b. React	18
c. Tailwind	19
d. PrimeReact	19
e. Material UI	20
IV. További fontos NPM csomagok	21

a.	react-responsive-carousel	21
b.	react-router-dom	21
c.	axios	22
<b>V.</b>	<b>Jogosultságok</b>	<b>23</b>
<b>VI.</b>	<b>Oldalak és funkciók</b>	<b>23</b>
a.	Regisztráció	23
b.	Bejelentkezés	24
c.	Jelszóváltoztató	24
d.	Szűrés	24
e.	Fizetés	25
f.	Visszajelzés	25
g.	Adatváltoztatás	25
h.	Context	26
<b>5.</b>	<b>Csapatmunka</b>	<b>27</b>
I.	A kezdetek	27
II.	Discord	27
III.	Verziókezelés	28
<b>6.</b>	<b>Hosting</b>	<b>29</b>
<b>7.</b>	<b>CareCrop mint szimbólum</b>	<b>29</b>
I.	A logó	29
II.	Ki nem választott tervezetek	29
<b>8.</b>	<b>Források</b>	<b>31</b>

# 1. A vizsgaremekről

## I. Mi nekünk a CareCrop?

A CareCrop egy olyan mezőgazdasági problémára talál megoldást, ami, a csapattagok véleménye szerint, a technológiai fejlődés mértékét figyelembe véve kikerülhetetlen. A CareCrop egy olyan egységes piaci teret biztosít mind magánszemélyek, mind mezőgazdasági cikket árusító cégek számára, amivel megnyílik az online kereskedelmi tér a gazdálkodók előtt.

## II. Mi is a probléma?

A csapat tagjai nyári munkatapasztalat szerzése közben megfigyelhették, hogy a mezőgazdasági cikkekkel foglalkozó kereskedés, az vagy papíron folyik, vagy már elavult kezelőprogramokban, külön ezen gazdasági ágazat részére szabott független webshop még nem létezik. Ezért gondolta azt csapatunk, hogy betömné ezt a piaci rést egy olyan apróhirdetéseket oldal elkészítésével, ami a gazdák nagy segítségére lehet.

## III. Miért CareCrop?

Ez az egyszerű szó egybefoglalja a webshop működésének két mozzatát: A *Care* a növényvédő vegyszerekre hivatkozik, míg a *Crop* magukra a növényekre. Valamint azért, mert a CropCare kifejezés már le lett védve és nem akarunk a jövőben jogi vitákba keveredni egy műtrágyaárusító céggel.

## IV. Mi várható a jövőben?

Igaz, hogy kiemeltük az oldalunk függetlenségét, miszerint nem egy külön megrendelő kérésére készítettük, a nemrég megjelent rendelkezéseket figyelembe véve a jövőben mindenképpen törekedni fogunk egy, az ágazatban ismert, céggel



felvenni a kapcsolatot. Hosszútávú terveinkbe beletartozik még az internacionalizáció is, főként a német nyelv implementálása.

## 2. Adatbázis

### I. Tervezés

Az ötlet meghatározása után, miszerint a mezőgazdasági kereskedelmet szeretnénk megkönnyíteni, a csapat első feladata az adatbázisterv felállítása volt. Mindenképpen olyan adatbázist képzeltek el, és később valósítottunk meg, ami minden probléma nélkül tudja tárolni az átadott adatokat és tökéletesen megfelel az előírt elvárásoknak. Abban a kezdeteknél sikerült megállapodnunk, hogy az



adatbázis MySQL adatbázis-kezelőt fog alkalmazni, mivel a csapat minden tagjának szimpatikusabb volt a használata, mint a másik elérhető opció, a NoSQL. Végül kétféle modellt tervezünk: a draw.io alkalmazás segítségével egy ER-modellt, valamint a dbForge-ban elérhető funkcióknak hála egy AB-moddellel is sikerült vizualizálnunk elképzeléseinket.

### II. draw.io

A draw.io is a JavaScript-ben írt, kliensoldali tervezőprogram, amit főként diagrammok készítésére használnak. A program fejlesztését a brit David Benson és a svájci Gaudenz Alder kezdte el még 2002-ben JGraph néven. Választásunk azért esett a draw.io szolgáltatására, mert nemcsak átlátható és könnyen kezelhető a felülete, de nagy szerencsénkre az idei évi tanulmányaink során is megtapasztalhattuk használatának előnyeit.

### III. MariaDB/dbForge

Mint korábban említettem, MySQL kezeléséhez a dbForge nevű program mellett döntöttünk. A programot a Devart készíti, akiknek a célja, hogy gyors,

effektív és grafikusan átlátható felületet biztosítsanak az SQL Szerverrel, MySQL-lel, PostgreSQL-lel és az Oracle-lel való munkához. A mi esetünkben természetesen csak a MySQL része volt fontos a programcsomagnak.

A saját backend szerverünkön elengedhetetlen szerepet tölt be a MariaDB használata. A Debian alapú szerveren egy MySQL script lefuttatása után sikeresen tudtuk kezelni az adatbázist pontosan olyan precizitással mintha a lokális gépen használtunk volna dbForge-ot. Egyetlen hátulütője a MariaDB-nek a szöveges kezelési felület lehet, viszont mivel a parancsok megegyeznek bármilyen más MySQL-t kezelő program parancsaival, a kezelés nem okozott gondot a csapat egyik tagjának sem. A használatáról és a hitelesítésről a Backend taglaló fejezetben tudhat meg többet.



```
root@6863:~# mysql -u mysql -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 903
Server version: 10.11.6-MariaDB-0+deb12u1 Debian 12

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use carecrop
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [carecrop]> select * from users;
```

*A MariaDB kezelése saját szerveren*

## IV. Táblák

### a. Termelők

A termelők táblában azokat a személyeket tároljuk el, akik regisztráltak az oldalra és így a vásárláson kívül a feltöltés és a visszajelzés lehetősége is fennáll nekik.

id	INT
termeny_id	INT
lakcím	VARCHAR
név	VARCHAR
jogosultság	VARCHAR
telefonszám	VARCHAR
számlaadatok	VARCHAR
Constraints	
Indexes	

### b. Telephelyek

A telephelyek táblában lesznek a jövőben eltárolva a velünk kapcsolatban álló cégek telephelyei, akik által gördülékenyen mehet a zöld könyves növényvédő szerek árusítása is.

id	INT
cím	VARCHAR
név	VARCHAR
telepvezető	VARCHAR
telefonszám	VARCHAR
termeny_id	INT
jogosultság	VARCHAR
Constraints	
Indexes	

### c. Termények

id	INT
szezon	VARCHAR
megnevezés	VARCHAR
egységár	INT
minőség	VARCHAR
Constraints	

A termények táblában találhatóak az apróhirdetések lényeges adatai. A termés szezonja, neve, a hirdetés ára és a termék minősége, ami főleg búzát tartalmazó hirdetésnél képvisel nagy jelentőséget.

## d. Vásárlás

vásárlás	
id	INT
termény_id	INT
vásárló_id	INT
megjegyzés	VARCHAR
ár	INT
fizetés	VARCHAR
mennyiség	VARCHAR
Constraints	
Indexes	

A vásárlás táblában található adatok mind egy adott tranzakcióhoz köthetőek. Amikor a vásárlás befejeződik, akkor kapja meg az adatbázis az adatokat a frontendtől, addig a kosár tartalma a localStorage-ban tárolódik.

## e. Vásárlók

vásárlók	
id	INT
felhasználónév	VARCHAR
név	VARCHAR
email	VARCHAR
jelszó	VARCHAR
telefonszám	VARCHAR
jogosultság	INT
regisztráció	DATE
Constraints	

A vásárlók táblában található minden olyan adat azokról a felhasználóktól, akik folytattak le vásárlást. A lényeges mező a táblában a *jogosultság* nevet viseli: Ha a vásárló regisztrált, 1-es értéket vesz fel, ha csak egy vendég, akkor 0-s értéket.

## f. Support

A support táblában tároljuk a webshop visszajelzés funkciójának lényeges adatait. A megadott üzenet szöveg típusú adat, míg a lenyíló listából kiválasztott típus értékét egy szám adattípusban raktározzuk el. (A funkció csak regisztráció és bejelentkezés után érhető el)

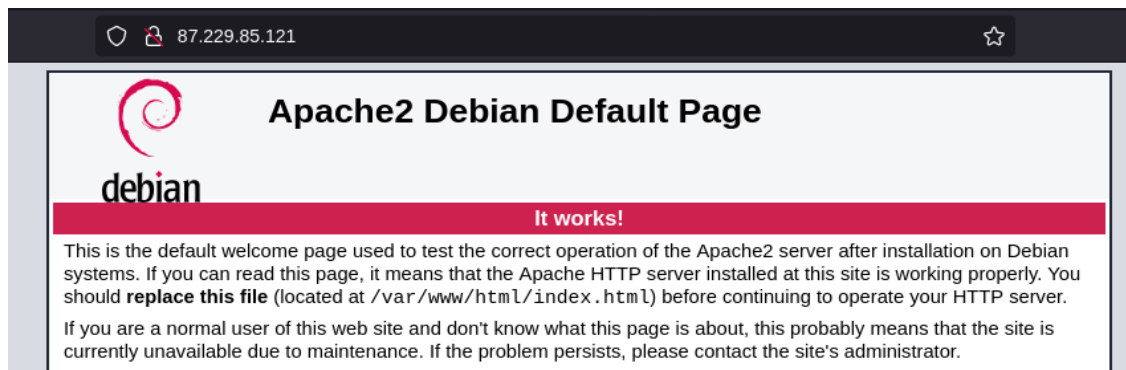
support	
id	INT
vásárló_id	INT
üzenet	VARCHAR
típus	
Constraints	
Indexes	

## 3. Backend

### I. LAMP

A backendhez a **LAMP** stacket használtuk, ami a következőképpen néz ki:

- **Linux:** Debian 12 operációs rendszer
- **Apache:** Apache 2 webserverv
- **MariaDB:** Adatbázis kezelő rendszer
- **PHP:** Natív PHP 8, dinamikus tartalmat feldolgozó backend nyelv



```
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_SESSION["username"]))
{
    $sql = "SELECT COUNT(id) FROM products;";
    $id = array_values($conn->query($sql)->fetch_assoc())[0];

    $sanitized_season = trim(htmlspecialchars($_REQUEST["season"]));
    $sanitized_name = trim(htmlspecialchars($_REQUEST["name"]));
    $sanitized_quality = trim(htmlspecialchars($_REQUEST["quality"]));
    $sanitized_price = (int)trim(htmlspecialchars($_REQUEST["price"]));

    $sql = "INSERT INTO products VALUES ('" . $id . "', '" . $sanitized_season . "', '" . $sanitized_name . "', '" . $sanitized_quality . "', '" . $sanitized_price . "')";
    try {
        $conn->query($sql);
    } catch (Exception $e) {
        echo '{"success": false}';
        http_response_code(400);
        die();
    }

    echo '{"success": true}';
}
```

*A webserverv kezdőoldala és natív PHP kód*

## II. A backend felépítése

A backendhez natív PHP 8-at használunk, a mysqli (*MySQL Improved Extension*) kiegészítővel. Harmadik féltől származó könyvtárat/kiegészítőt/keretrendszert nem használtunk.

```
$sql = "SELECT COUNT(id) FROM products;";  
$id = array_values($conn->query($sql)->fetch_assoc())[0];
```

### a. Hitelesítés a MariaDB adatbázishoz

A MariaDB adatbázishoz szükséges hitelesítést úgy oldottuk meg, hogy minden PHP fájl elejére beillesztettük az *auth.php* nevű fájlt, amiben a hitelesítés adatai találhatóak, a biztonság miatt ez a fájl a .gitignore része, tehát a backend repository-ba nem kerül feltöltésre.

```
<?php  
$server = "127.0.0.1:3306";  
$username = "ide jön a név";  
$password = "ide jön a jelszó";  
$database = "carecrop";  
  
$conn = new mysqli($server, $username, $password, "carecrop");  
?>
```

### b. Végpontok működése

Legtöbb esetben a végpontok egymáshoz hasonlóan működnek: először megtisztítjuk a felhasználó által bevitt adatokat és ha szükséges, elvégezzük a típuskonverziót, majd a kérés típusa alapján adatbázis kezelési műveleteket végzünk el.

```
$sanitized_note = trim(htmlspecialchars($_REQUEST["note"]));
$sanitized_price = (int)trim(htmlspecialchars($_REQUEST["price"]));
$sanitized_product = trim(htmlspecialchars($_REQUEST["product"]));
$sanitized_quantity = (int)trim(htmlspecialchars($_REQUEST["quantity"]));
```

```
else if ($_SERVER["REQUEST_METHOD"] == "GET")
{
    $sql = "SELECT * FROM products;";
    $results = array();
    try {
        $result = $conn->query($sql);
    } catch (Exception $e) {
        echo "{\"success\": false}";
        http_response_code(400);
        die();
    }
    while($row = $result->fetch_assoc())
        array_push($results, $row);
    echo json_encode($results, true);
}
```


A backend ezután visszaad egy választ, ami JSON formátumban lesz. Ezt a formátumot a fejlécben kényszerítjük rá a válaszra.

```
header("Content-Type: application/json");
```

Sikeres GET kérések esetén a kívánt adatokat küldjük vissza, míg más sikeres kérelmek (POST, DELETE, PUT) esetén egy bool típusú változót küldünk az elvégzett művelet sikerességéről:

JSONRaw DataHeaders

SaveCopyCollapse AllExpand All

 Filter JSON

id:"0"

username:"test\_name4"

email:"test@test.com"

admin:"0"

registration\_date:"2024-04-06"

```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
success: true
```

### III. Hibakezelés

Hibás kérések esetén a megfelelő hibaüzeneteket küldünk vissza a frontend felé a kapcsolódó státusz kóddal ellátva. Az alább látható példák a *register.php* és a *support.php* felé intézett POST kérések hibakódjai és üzenetei láthatóak.

```
400 POST localhost:8000 register.php
```

```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
error: "Érvénytelen adatok!"
```

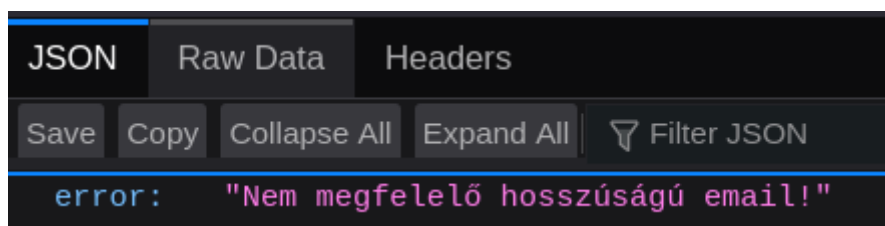
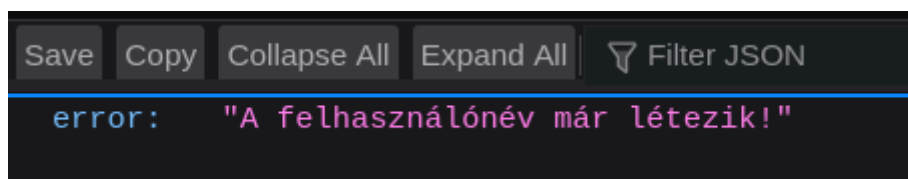
```
401 GET localhost:8000 support.php
```

```
JSON  Raw Data  Headers
Save Copy Collapse All Expand All Filter JSON
error: "Nincs bejelentkezve!"
```



## IV. Regisztráció

A regisztráció folyamán először megtisztítjuk a beérkező adatokat, majd azt nézzük meg, hogy van-e duplikált e-mail vagy felhasználónév, illetve az adatok hosszát is megvizsgáljuk. A képeken az email-cím és a felhasználónév ellenőrzésének hibakódjai láthatók. Amennyiben minden rendben van, az adatokat mentjük az adatbázisban.



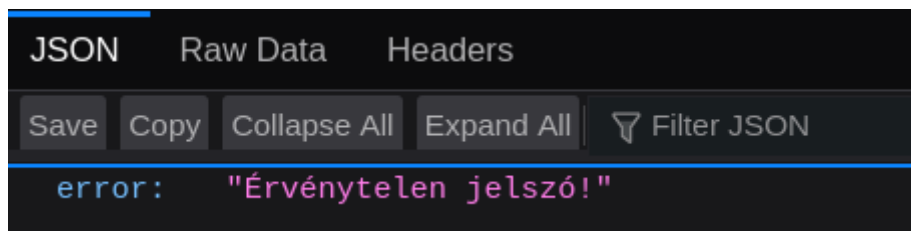
A jelszót a *Crypt Blowfish* nevű algoritmussal hasheljük és hogy ne lehessen az adatbázisban hozzáférni felhasználók jelszavaihoz, így csak a hash-elt jelszavakat tároljuk el.

```
MariaDB [carecrop]> SELECT password FROM users WHERE id = 0;
+-----+
| password |
+-----+
| $2y$10$.ldC4QQ.gN8SnJky1CkStumKGcDAXuYXCemzI3BMAZoxzYSPawY.m |
+-----+
1 row in set (0.000 sec)

MariaDB [carecrop]> █
```

## V. Bejelentkezés

A kód először ellenőrzi, hogy a felhasználó be van-e már jelentkezve. Ha igen, hibaüzenetet küld. Ha a felhasználó még nem jelentkezett be, a kód ellenőrzi, hogy a megadott felhasználónév létezik-e az adatbázisban. Ha a felhasználónév létezik, a kód ellenőrzi, hogy a megadott jelszó illeszkedik-e a tárolt jelszó kódjával.



A sikeres bejelentkezést követően, a felhasználó adatait a PHP munkamenetben (\$\_SESSION) tároljuk. Minden fájlban lévő felhasználó hitelesítést ezzel a munkamenet adatokkal végzünk.

```
$sql = "SELECT * FROM users WHERE username = '" . $sanitized_name . "'";
$user_data = $conn->query($sql)->fetch_assoc();
$hashed_password = $user_data["password"];

if (password_verify($_REQUEST["password"], $hashed_password))
{
    foreach ($user_data as $key => $value)
    {
        $_SESSION[$key] = $value;
    }

    echo "{\"success\": true}";
}
else
{
    echo "{\"error\": \"Érvénytelen jelszó!\"}";
    http_response_code(400);
}
```

## VI. Jogosultságok

Új adatok felvételéhez (POST) és törléséhez (DELETE) a felhasználónak muszáj regisztrálni majd belépni. Legtöbb esetben azonban, az adatok lekéréséhez (GET) a vendégfiókok (guest) is hozzáférnek bármilyen hitelesítés nélkül (telephelyek, termények stb.). Ez alól kivétel a fizetés után leadott kosarak adatai, amiket csak adminisztrátor hozzáféréssel lehet elérni. Az alábbi képeken a jogosultságkezelés látható.

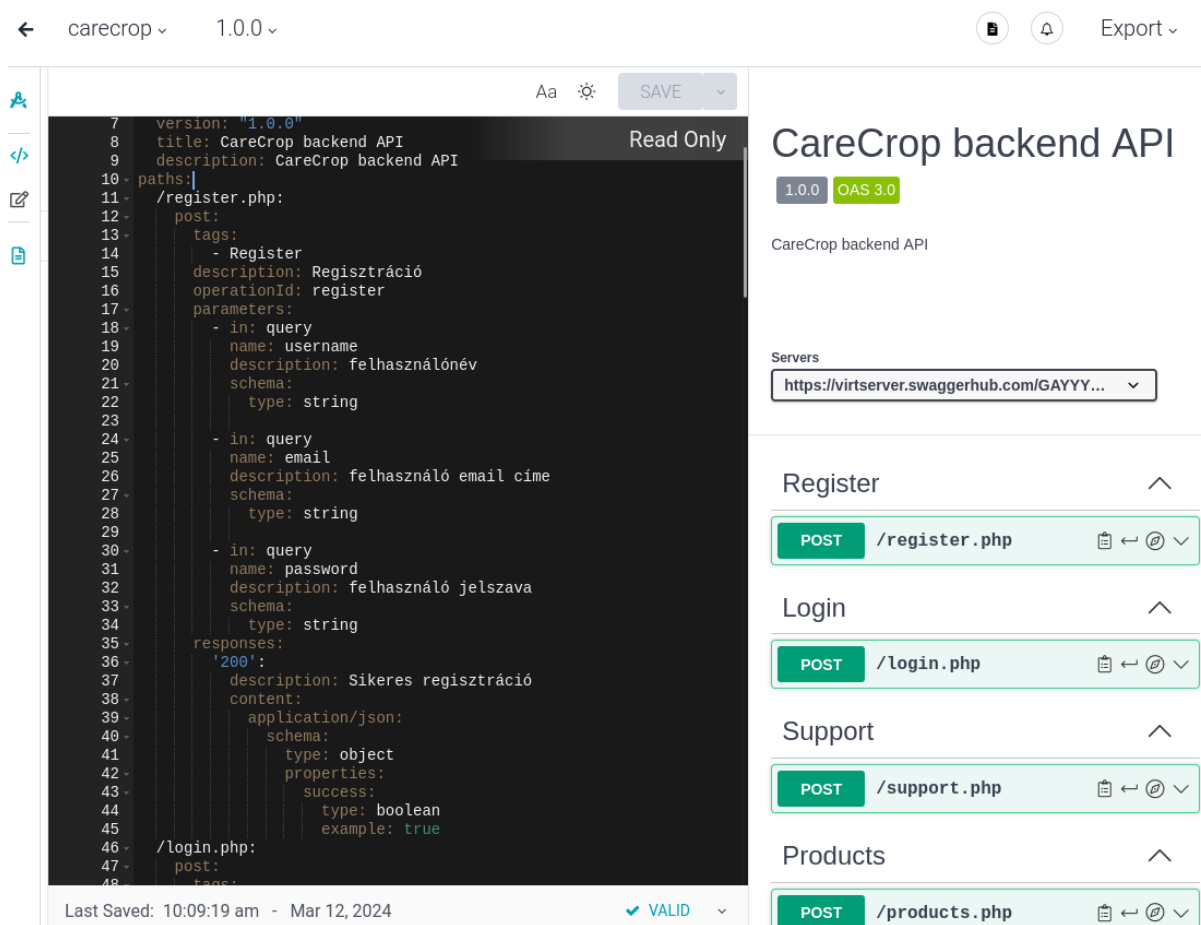
```
if (!isset($_SESSION["username"]))
{
    echo "{\"error\": \"Nincs bejelentkezve!\"}";
    http_response_code(401);
    die();
}
```

```
else if ($_SERVER["REQUEST_METHOD"] == "GET"
&& isset($_SESSION["admin"]))
{
    $sql = "SELECT * FROM cart;";
    $results = array();
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST"
&& isset($_SESSION["username"]))
{
    $sql = "SELECT COUNT(id) FROM cart;";
    $id = array_values($conn->query($sql)->fetch_assoc())[0];
```

## VII. Swagger

Ezen dokumentáció mellett a backendhez készült egy külön Swagger dokumentáció is. A Swagger segítségével a szerverre feltöltés előtt tudtuk kezelni az adott végpontok helyes vagy hibás működését, valamint a visszaérkező válaszok pontosságát. Az alábbi képeken a fő felület és egy teszt látható.



The image shows the Swagger UI for the CareCrop backend API. The left sidebar contains a list of endpoints: `/register.php`, `/login.php`, `/support.php`, and `/products.php`. The main area displays the details for the `/register.php` endpoint, which is a POST request. The API definition is shown in a code editor, and the right sidebar provides a summary of the endpoint, including its tags, description, and parameters.

```
7 version: "1.0.0"
8 title: CareCrop backend API
9 description: CareCrop backend API
10 paths:
11   /register.php:
12     post:
13       tags:
14         - Register
15       description: Regisztráció
16       operationId: register
17       parameters:
18         - in: query
19           name: username
20           description: felhasználónév
21           schema:
22             type: string
23         - in: query
24           name: email
25           description: felhasználó email címe
26           schema:
27             type: string
28         - in: query
29           name: password
30           description: felhasználó jelszava
31           schema:
32             type: string
33       responses:
34         '200':
35           description: Sikeres regisztráció
36           content:
37             application/json:
38               schema:
39                 type: object
40                 properties:
41                   success:
42                     type: boolean
43                     example: true
44   /login.php:
45     post:
46       tags:
47         - Login
```

Register

POST /register.php

Login

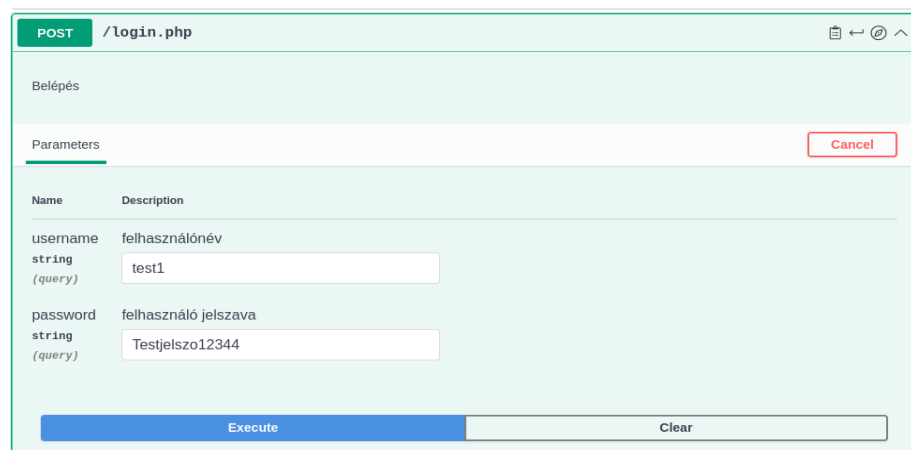
POST /login.php

Support

POST /support.php

Products

POST /products.php



















The image shows the Swagger UI test interface for the `/login.php` endpoint. The interface is titled "Belépés" (Login) and includes a "Parameters" section. The parameters are listed in a table with columns for "Name", "Description", and "Value". The "username" parameter is a query parameter of type "string" with a value of "test1". The "password" parameter is a query parameter of type "string" with a value of "Testjelszo12344". There are "Execute" and "Clear" buttons at the bottom.

Name	Description	Value
username	felhasználónév	test1
password	felhasználó jelszava	Testjelszo12344

Execute Clear

## VIII. Git commitok

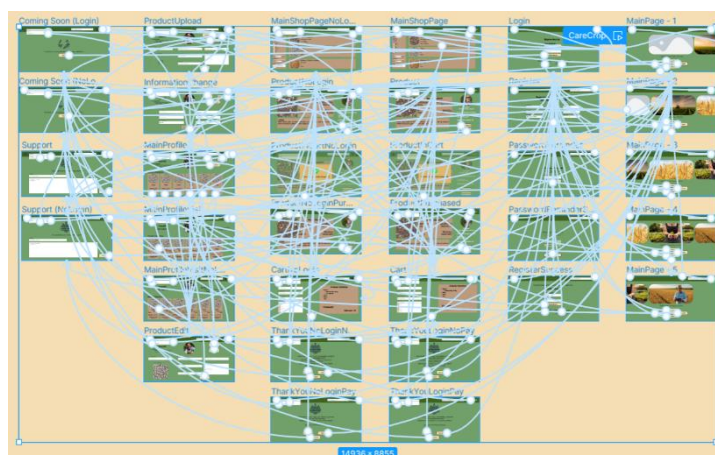
Mind a backend és a frontend verziókezelését GitHubon oldottuk meg. A commit üzeneteknek igyekeztünk beszélő neveket adni, a lényeg az volt, hogy mindig utaljon az elvégzett munkára és annak eredményére.

<b>added user data and change password end points, added status codes to responses</b> gayymess committed 2 days ago	84c8e43		
Commits on Apr 12, 2024			
<b>Updated login CORS</b> BalintHejner committed 3 days ago	Verified b8e2a12		
<b>added logout</b> gayymess committed 3 days ago	8cb3602		
<b>added error handling for SQL queries and added length check for register</b> gayymess committed 3 days ago	595d182		
Commits on Apr 6, 2024			
<b>headers added for solving cors issues</b> Bálint Hejner committed last week	c22ee43		
<b>added GET and DELETE handling for bases and manufacturers, fixed some errors</b> gayymess committed last week	b7402df		
Commits on Mar 31, 2024			
<b>added bases</b> gayymess committed 2 weeks ago	61ab72a		
<b>added manufacturers and bases</b> gayymess committed 2 weeks ago	b8c62eb		

## I. Tervezés

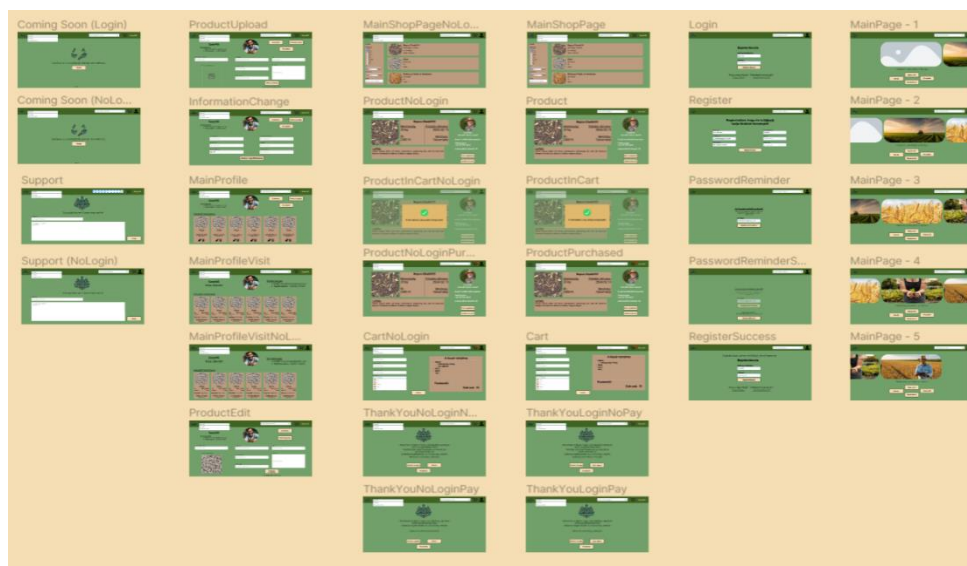
Mint minden webshopnál, a miénknél is nagyon fontos volt megjelenés és a dizájn szerepe, hiszen az egyszerű felhasználót azzal lehet legjobban megnyerni. A munka megkezdése előtt egyöntetű döntés született arról, hogy mielőtt megkezdődne a fejlesztése a frontendnek, mindenképpen időt kell szánni az oldal megtervezésére. A csapat ennek a feladatnak az elvégzésére a Figma névre hallgató alkalmazást használta.

A Figma egy csapatmunkán alapuló webalkalmazás tervezéshez, további offline funkciókkal, amelyeket a macOS és Windows asztali alkalmazásai tesznek lehetővé. A Figma szolgáltatáskészlete a felhasználói felület és a felhasználói élmény kialakítására összpontosít, hangsúlyt fektetve a valós idejű együttműködésre. Választásunk azért esett a Figma-ra mint tervezőplatform, mert a többi, Figma-hoz hasonló tervezőprogramot nem találtuk annyira szabadon kezelhetőnek, mint a Figma-t. A választásban segített a tény is, hogy a tervezéskor egy másik csapat az osztályban már pozitív használati élményekkel rendelkezett, amit meg is osztott velünk.



*Az ún. **navigációs háló**. Az összes navigációs lehetőség látható az ábrán*

A 1.5 hónapot igénybe vevő tervezési folyamat alatt a Figma segítségével nem csak az oldal kinézetét sikerült megtervezni, hanem az egyéb hasznos funkcióknak hála az oldalak közötti navigáció felvázolása is sikeresen megtörtént.



*A Figma felületén az előzetes tervek*

### III. Fejlesztői környezetek

#### a. JavaScript

A frontend fejlesztés JavaScript nyelven történt. Ez a web alaptechnológiájának számít. Sokat gondolkodtunk a JavaScript és ennek típusos változata, a TypeScript között. Bár a TypeScript típusai segítenek a szintaktikai hibák elkerülésében, így csökkentve a futási problémákat, végül a letisztultság miatt a JavaScript-re esett a választásunk.

#### b. React

A React egy ingyenes és nyílt forráskódú JavaScript-könyvtár, amely frontend fejlesztésére szolgál. Alkotója a Meta-csoport, de a fejlesztést velük együtt a felhasználói közösség végzi. A projekt fejlesztése alatt a függvény-alapú Reactot használtuk alapoknak.

A React használata lehetővé tette a .jsx kiterjesztésű állományok használatát. Az alábbi képen egy tipikus React kódot láthatunk, esetünkben egyik gomb-komponensünk kódját:

```
export default function Button2({ text, className, click }) {  
  const navigate = useNavigate()  
  const path = () => {  
    navigate(click)  
  }  
  
  return (  
    <main className="flex flex-col items-center justify-center">  
      <button className={className} onClick={path}>  
        {text}  
      </button>  
    </main>  
  );  
}
```

## C. Tailwind

A Tailwind CSS egy utility-first CSS (Cascading Style Sheets) keretrendszer előre definiált osztályokkal, amelyek segítségével közvetlenül a markupban készíthetünk és tervezhetünk weboldalakat. Ez a csomag lehetővé teszi, hogy CSS-t írjon a HTML-ben előre meghatározott osztályok formájában. Használatával felgyorsítottuk az oldalak elkészítésének a sebességét, további bővítések esetén is gyorsabban tudjuk fejleszteni alkalmazásunkat.



## d. PrimeReact



A PrimeReact a felhasználói felület komponenseinek legteljesebb készlete, amely több mint 80 előre elkészített React komponenst nyújt. Ha mégse tudunk választani a különféle előre elkészített témák közül, a PrimeReact lehetőséget nyújt az



általunk választott által választott CSS-könyvtárral, például a Tailwind CSS-sel díszíteni. Legnagyobb segítséget az űrlapok egyes elemeinél nyújtotta.

## e. Material UI

A Material Ui a PrimeReact mellett egy másik könyvtár, amit használtunk. Ennek az volt az oka, hogy a Figma dizájnban megtervezett oldalak kinézetéhez hűek akartunk maradni. Bár az előre leprogramozott komponensek testreszabhatóak, mivel nem voltunk elégedettek az eredménnyel, ezért vontunk be több frontend könyvtárat is. A Material UI Grid komponense nyújtotta talán a legnagyobb segítséget a projekt elkészítése alatt. A Grid-nek hála egyes oldalakon nagyon könnyen volt kivitelezhető a teljes reszponzivitás.



```
<Grid container spacing={7} style={{marginLeft: 'auto', marginRight: 'auto'}}>
  <Grid item xs={10} sm={8} md={6} lg={3.6} xl={4}>
    <Input type="text"
      placeholder="Teljes név" change={handleNameChange} />
    <div>
      <Input type="text" placeholder="Felhasználónév"
        change={handleUsernameChange} />
    </div>
  </Grid>
  <Grid item xs={10} sm={8} md={6} lg={3.6} xl={4}>
    <Input type="email"
      change={handleEmailChange} placeholder="E-mail cím" />
    <div>
      <Input type="tel" change={handlePhoneChange}
        pattern="[0-9]{2}-[0-9]{2}-[0-9]{3}-[0-9]{4}" placeholder="Telefonszám" />
    </div>
  </Grid>
  <Grid item xs={10} sm={8} md={6} lg={3.6} xl={4}>
    <Input type="password" name="password"
      change={handlePasswordChange} placeholder="Jelszó" />
    <div>
      <Input type="password" name="confirmedPassword"
        change={handlePasswordChange} placeholder="Jelszó megerősítése"/>
    </div>
  </Grid>
</Grid>
```

## IV. További fontos NPM csomagok

### a. react-responsive-carousel

A főoldalon szereplő carousel megoldásához több opciót is kipróbáltunk (például a PrimeReact által biztosított komponenst is), de végül úgy döntöttünk, hogy a reszponzivitást szem előtt tartva a react-responsive-carousel csomagot használjuk fel. Egyik legnagyobb megnyerő tulajdonsága az autoplay és az interval property-páros volt, ami lehetővé teszi, hogy beavatkozás nélkül, folyamatosan tudjon a carousel működni amíg a felhasználó a főoldalon tartózkodik. (Jelen esetünkben a carousel 1 képen 2 másodpercet tölt el váltás előtt).

```
<Carousel showThumbs={false}
showStatus={false}
className="rounded-full"
showIndicators={false}
showArrows={false}
infiniteLoop={true}
autoplay={true}
interval={2000}
stopOnHover={false}
transitionTime={500}>
  {
    images.map((image, index) => (
      <div key={index}>
        <img loading="lazy" src={image.url} className="carousel-image" />
      </div>
    ))
  }
</Carousel>
```

### b. react-router-dom

A react-router-dom a React JS-ben egy JavaScript-csomag, amely lehetővé teszi a React használatával bonyolult kliensoldali alkalmazások létrehozását. A kezdetben 2013-ban elindított program a mai online alkalmazások egyik legjelentősebb útválasztó könyvtárává vált. A React Router egy JavaScript keretrendszer, amit a kliens és szerver-oldal közötti routing megvalósítására használtunk.

Ennek a JavaScript keretrendszernek a segítségével oldottuk meg a weboldalak közötti navigációt.

```
const navigate = useNavigate()
const path = () => {
  navigate(click)
}
```

E csomag lehetővé tette számunkra, hogy egyszerűen kezeljük a URL-eket és az alkalmazásunk státuszát. Az alkalmazásban specifikáljuk az összes lehetséges URL sablont, és megadjuk, hogy adott végponthoz melyik UI komponensek kerüljenek megjelenítésre. Ezzel csökkentjük a kód mennyiségét, javítva az alkalmazás átláthatóságát és karbantarthatóságát.

```
<ContextProvider>
  <BrowserRouter>
    <Routes>
      <Route path="/" element={<MainPage/>}/>
      <Route path="/login" element={<LoginPage/>}/>
      <Route path="/register" element={<RegisterPage/>}/>
      <Route path="/support" element={<SupportPage />}/>
      <Route path="/passwordchange" element={<PasswordChangerPage/>}/>
      <Route path="/cart" element={<CartPage/>}/>
      <Route path="/product" element={<ProductViewPage/>}/>
      <Route path="/productedit" element={<ProductEditPage/>}/>
      <Route path="/productupload" element={<ProductUploadPage/>}/>
      <Route path="/comingsoon" element={<ComingSoonPage/>}/>
      <Route path="/useredit" element={<InformationChangePage/>}/>
      <Route path="/thankyou" element={<ThankYouPage/>}/>
      <Route path="/shop" element={<MainShopPage/>}/>
      <Route path="/userprofile" element={<ProfilePage/>}/>
      <Route path="/profile" element={<OwnProfilePage/>}/>
    </Routes>
  </BrowserRouter>
</ContextProvider>
```

### C. axios

A backenddel való kommunikációhoz az axios csomagot használtuk a már React-ben szereplő fetch-csel szemben. A csomag segítségével egy könnyen kezelhető, egyszerű szintaxisal rendelkező parancsot kaptunk, mint ahogy példánkon is látható.

```
const response = await axios.get(
  `login.php?username=${username}&password=${password}`,
  { username: username, password: password },
  { withCredentials: true }
);
```

## V. Jogosultságok

- Egyszerű felhasználó (guest)
- Gazda / Regisztrált felhasználó

Az alkalmazást lehet használni bejelentkezés vagy regisztráció nélkül is, ilyenkor a felhasználónak jogosultsága van a weboldalon böngészni, terményeket nézni, és ezeket a terményeket megvásárolni. Ilyenkor a felhasználónak meg kell adnia adatait sikeres vásárlás előtt. Ha a felhasználó eladni is akar az alkalmazásban, akkor regisztrálni, illetve bejelentkezni kell. Ilyenkor, ha megadja a megfelelő adatokat, akkor felkerül hirdetése a weboldalra. A bejelentkezett felhasználó saját hirdetéseit meg tudja nézni, szerkeszteni tudja őket.

## VI. Oldalak és funkciók

### a. Regisztráció

**Regisztráljon, hogy ön is Nálunk tudja hirdetni terményeit!**

<input type="text"/>	<input type="text"/>	<input type="text"/>
Teljes név	E-mail cím	Jelszó
<input type="text"/>	<input type="text"/>	<input type="text"/>
Felhasználónév	Telefonszám	Jelszó megerősítése

**Regisztráció**

A felhasználó ide navigálva a főoldalról vagy a bejelentkezésről saját fiókot tud létrehozni. Itt meg kell adnia a nevét, e-mail címét, illetve telefon számát. Ezután megadja a felhasználónevét és jelszavát. Ha minden adat megfelel a szabványoknak, és a megadott jelszó is megegyezik a jelszóellenőrzéssel, akkor a regisztráció gombra kattintva létre jön a felhasználó saját fiókja.

## b. Bejelentkezés



**Bejelentkezés**

Felhasználónév:

Jelszó:

**Bejelentkezés**

Nincs még fiókja? [Regisztráljon](#)

Elfelejtette jelszavát? [Jelszóemlékeztető](#)

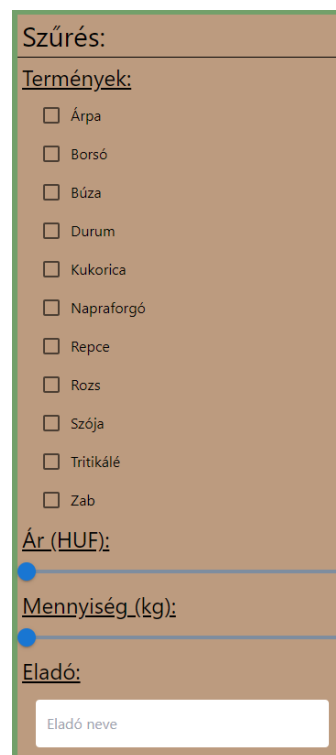
A felhasználó a főoldaltól egy gombnyomással el tud navigálni a bejelentkezés felületre. A felhasználónevét és jelszavát megadva elérhetővé válik számára az alkalmazás összes funkciója. Abban az esetben, ha a felhasználó még nem rendelkezik fiókkal akkor a Regisztráció gombra kattintva új fiókot hozhat létre.

## c. Jelszóváltoztató

Ha esetleg a felhasználó szeretné megváltoztatni megadott jelszavát, biztosítottunk számára egy változtató lehetőséget, amit a Bejelentkezés oldaláról lehet elérni. Ezen az oldalon az e-mail címének, valamint a régi és az új jelszónak megadásával a felhasználó egy gombnyomásra frissíti régi jelszavát az adatbázisban.

## d. Szűrés

Alapfunkciója egy jó webshopnak az adott feltételek alapján való szűrése az elérhető termékeknek. Ez nálunk is elérhető természetesen: A feltöltött apróhirdetéseket a megadott termék, az ár, a mennyiség, valamint az eladó neve alapján lehet szűrni.



**Szűrés:**

Termények:

- ☐ Árpa
- ☐ Borsó
- ☐ Búza
- ☐ Durum
- ☐ Kukorica
- ☐ Napraforgó
- ☐ Repce
- ☐ Rozs
- ☐ Szója
- ☐ Triticálé
- ☐ Zab

Ár (HUF):

Mennyiség (kg):

Eladó:

## e. Fizetés

Ha a felhasználó készen van a vásárlással, akkor a fejlécen található bevásárlókocsi ikonra kattintva eljuthat a fizetési felületre, ahol egy rövid űrlapot kitöltve már el is kezdődhet a fizetési folyamat, valamint láthatja kosarának teljes tartalmát.

## f. Visszajelzés

A fizetési folyamat befejeződése után a felhasználónak lehetősége van visszajelzést küldeni. Egy legördülő listából ki lehet választani a visszajelzés típusát majd azt, és a megírt üzenetet egy gombnyomásra elküldi a backendnek tárolásra.





*Visszajelzése van? Ossa meg velünk!*

Visszajelzés típusa ▼ Küldés

Visszajelzés

## g. Adatváltoztatás

A weboldalon külön lehetőséget biztosítunk mind a felhasználó adatainak, mind az általa feltöltött termékek adatainak megváltoztatására is.

## h. Context

Az alkalmazás fejlesztéséhez használunk egy saját Contextet is, mely a globális állapotkezelést valósítja meg az alkalmazásban. Ahhoz, hogy Reactban egy komponens hozzáférjen egy globális változóhoz, egy Providerbe kell tenni.

Mi a Routert raktuk bele egy Providerbe, így a Router mint a Provider leszármazottja hozzáfér a Contexthez.

```
import { createContext } from "react";
import { useState } from "react";

const Context = createContext();

export function ContextProvider({ children }) {
  const [loggedIn, setLoggedIn] = useState(false);
  const [username, setUsername] = useState('');
  const [token, setToken] = useState('');
  const [cart, setCart] = useState([]);
  const [wishlist, setWishlist] = useState([]);
  const [categories, setCategories] = useState([]);
  const [products, setProducts] = useState(true);
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');
  const [phone, setPhone] = useState('');

  const contextValues = { ...
  };

  return (
    <Context.Provider value={contextValues}>
      {children}
    </Context.Provider>
  )
}

export default Context;
```

## 5. Csapatmunka

### I. A kezdetek

A projekt elkezdése előtt fontosnak tartottuk a feladatok megfelelő elosztását. Úgy döntöttünk, hogy ketten fogunk frontend technológiával, és egy ember backenddel foglalkozni. Szó esett a full-stack webfejlesztésről is, de ezt az ötletet végül elvetettük. Így elszeparálva a feladatokat letisztultabban, gördülékenyebben tudunk dolgozni.

A csapat felépítése:

- Backend – Hidasi Zalán
- Frontend – Németh Csaba, Hejner Bálint
- Adatbázis – Hejner Bálint

### II. Discord

Iskolán kívül csapatunk legalább heti egyszer online is összeült. A Discord platformot használva, délutánonként csapatmegbeszéléseket tartottunk, szükség esetén akár többet is egy héten. A csapatmegbeszéléseknek nem csak azért volt hatalmas jelentősége, mert így tájékoztatni tudtuk a párhuzamosan futó munka fejleményeiről egymást, hanem azért is, mert egyik csapattársunk speciális tanrendben végzi tanulmányait, így vele ez volt az egyetlen jelentőségteljes kommunikációs forma a szöveges üzeneteken kívül.






### III. Verziókezelés

A csapatmunka megkönnyítése végett és a verziókövetés megvalósításához a GitHubot használtuk. Ezen a felületen nyomon követhető, ki, melyik feladatot végezte. Így biztonságban tudhattuk alkalmazásunkat, illetve tudtuk ellenőrizni egymás munkáját.

A Github felületén a projektet hoztunk létre, ehhez csapoltuk hozzá a megfelelő repository-kat. Ezekben a repository-kban külön branch-ekben dolgoztunk, melyekből egyet main-line munkára használtunk, majd ebből megfelelő működés esetén a került az új kódrészlet a release branch-be. A frontend részhez készült egy backup branch is, ha esetleg bármi történne a program branch-csel, ne vesszen kárba a munkánk. Különösen figyelünk a GitMoji-k használatára is, hogy a néha monoton munkába egy kis humort is tudjunk csempészni.

CareCrop							Add status update	✓	📄	⋮
Main (Table) Board Roadmap + New view										
Filter by keyword or by field Discard Save										
Title	Assignees	Status	Part beginning	Part End	Repository					
1 Backend - PHP #1	BalintHejner, dzsab...	Done	Jan 12, 2024	Apr 12, 2024	BalintHejner/CareCrop---Backend					
2 Database - MySQL #2	BalintHejner, dzsab...	Done	Dec 4, 2023	Feb 25, 2024	BalintHejner/CareCrop---Backend					
3 Frontend - React #1	BalintHejner and dz...	Done	Feb 26, 2024	Apr 14, 2024	BalintHejner/CareCrop---Frontend					

- ☐ ☒ **Routing**  
#8 by BalintHejner was closed 5 days ago
- ☐ ☒ **Gap between cards on MainShopPage**  
#5 by BalintHejner was closed 2 weeks ago
- ☐ ☒ **Cookies shouldn't reset when sending requests**  
#4 by BalintHejner was closed last week

 **Design and bug fixes**

## 6. Hosting

Csapatunk még a tervezési időszak legelején merész ötlettel állt elő: *Béreljünk saját szervert.* Az első pár Discordon tartott megbeszélésnek így ennek az ötletnek a kivitelezése volt a középpontja. Végül 2 hétnyi sikertelen próbálkozás után az Oracle felületén, egy magyar weboldalról sikerült 6 hónapra egy Debian 12 Standard operációs rendszerrel futó szervert bérelnünk (hostingbasis.hu). A szerver a bérlete óta rendületlenül fut és semmi problémát nem okozott a kezelése, hiszen a csapatban vannak olyan tagok, akik nagyon jól értenek a szöveges Linux disztribúciók használatához.

## 7. CareCrop mint szimbólum

### I. A logó

A csapat már az elején is megegyezett abban, hogy a *CareCrop* kifejezést nem csak, mint szó kell reprezentálni a projekt folyamán, hanem egy olyan módon is, ami könnyen felismerhető, és akár első látásra is megkülönbözteti bármi mástól. Így esett a választásunk egy logó elkészítésére, ami egyszerre tiszta képet ad a szolgáltatás lényegéről, valamint könnyen felismerhetővé teszi az oldalt.

Mivel a csapat egyik tagja sem rendelkezik sziporkázó grafikai készségekkel, úgy döntöttünk felhasználni a mesterséges intelligenciát, hogy készítsen a projekt célját méltóan képviselő logót.

### II. Ki nem választott tervezetek

Miután a Copilot által generált logókból kiválasztottuk a 3 legjobbat, pár napig gondolkodunk azon, melyik dizájn tudná legjobban képviselni azt a letisztult, de modern irányzatot, amit a weboldal dizájnjaival is követtünk.

Így jutottunk el a mai nap is használt logóhoz, de íme a másik kettő, ami épphogy lecsúszott a képzeletbeli dobogó felső fokáról:



*A 3. helyezett logó*



*A 2. helyezett logó*

## 8. Források

<https://axios-http.com>

<https://www.npmjs.com/package/react-responsive-carousel>

<https://www.npmjs.com/package/react-router-dom>

<https://primereact.org>

<https://mui.com>