**Имена:**  Александър Македонски, Васил Христов и Памела Славчева **фн:**  *8MI0800114, 5MI0800128, 1MI0800046***Начална година:** *2021* **Програма:** бакалавър, (КН) **Курс: 4**   
**Тема: 10.1  
Дата: 2025-01-28 Предмет: w23prj\_KN \_final имейл:** [**pamivladi@gmail.com**](mailto:pamivladi@gmail.com)

**преподавател:** доц. д-р Милен Петров

# ТЕМА: w23/10.1 ASCII art

## 1. Условие

Проектът **CharCraft** е вдъхновен от идеята да съчетае класическата естетика на текстовите символи с възможностите на съвременните уеб технологии, създавайки социална платформа, където потребителите могат да споделят, взаимодействат и вдъхновяват чрез ASCII изображения.

Целта на платформата е да осигури лесен и интуитивен начин за създаване, публикуване и коментиране на ASCII произведения, като същевременно насърчава активното участие на общността чрез функции като харесвания и коментари. Потребителите могат да качват своите творби, да получават обратна връзка от други ентусиасти и да изграждат своя дигитален архив от произведения.

CharCraft е разработен с мисъл за достъпност и удобство – от създаването на публикации до лесната навигация в платформата. Проектът включва интуитивен интерфейс, който позволява на потребителите да разглеждат и взаимодействат със съдържанието без излишни затруднения. Системата използва уеб технологии като **HTML, CSS и JavaScript**, комбинирани с мощта на **PHP** и **MySQL** за надеждно управление на данни и потребителски взаимодействия.

Функционалността на платформата е изградена върху принципа на асинхронна комуникация между клиентската и сървърната страна чрез **Fetch API**, което осигурява бързо зареждане и актуализация на съдържанието в реално време. Чрез механизми за динамично създаване на DOM елементи се постига гладка и интерактивна работа с публикациите, а потребителите могат да публикуват коментари и харесвания без нужда от презареждане на страницата.

Безопасността и удобството на потребителите също са ключови приоритети. Реализирани са мерки за валидиране на входните данни и осигуряване на коректно обработване на заявки, включително защита срещу неправомерен достъп.

Независимо дали потребителите са опитни артисти или просто любопитни ентусиасти, платформата предлага възможност да изразят себе си и да се свържат с общност, която споделя тяхната страст.

## 2. Въведение – извличане на изисквания

### 2.1 Роли

При анализа на изискванията на системата са идентифицирани следните основни роли на потребителите:

* **Гост (нерегистриран потребител)**
  + Преглед на публично достъпни публикации
  + Рисуване на ASCII изображение
  + Регистрация и влизане в системата
* **Регистриран потребител**
  + Създаване на ASCII публикации
  + Харесване и коментиране на публикации
  + Редактиране и изтриване на собствено съдържание
  + Управление на потребителския профил (промяна на парола)

### 2.2 Функционални изисквания

* **Управление на потребители:**
  + Регистрация на нови потребители чрез формуляр
  + Вход в системата с валидация на данни
  + Възможност за промяна на потребителски данни и парола?? // моля
  + Изход от системата
* **Управление на публикации:**
  + Добавяне на нови ASCII публикации с описание и заглавие
  + Преглед на съществуващи публикации
* **Интерактивност и комуникация:**
  + Динамично зареждане на публикации
  + Харесване и коментиране на публикации
* **Основни функции на инструмента за рисуване**
  + Интерактивно платно за рисуване:
    - Потребителят може да избира размер на платното (брой редове и колони).
    - Осигуряване на мрежова структура за лесно позициониране на символи.
    - Поддръжка на различни резолюции и адаптация към екрана на устройството.
  + Избор и поставяне на символи:
    - Потребителят може да избира символи от предварително дефиниран набор (ASCII символи).
    - Възможност за директно въвеждане на символи от клавиатурата.
    - Опция за бързо изчистване на определени области чрез изтриване на символи.
  + Цветови опции:
    - Поддръжка на цветни ASCII символи чрез избор на цвят и селектиране на цвят, който не е част от предварително зададените цветове.
  + Инструменти за рисуване:
    - Четка за ръчно добавяне на символи.
    - Гумичка за премахване на символи.
    - Инструмент за запълване на области със символи (кофа).
    - Избор на различни цветове и размери на символите.
  + Операции с платното:
    - Възможност за мащабиране (zoom in/out) на платното.
    - Придвижване (pan) в различни посоки за работа върху големи платна.
* **Съхраняване и споделяне на творби**
  + Запазване на творбата:
    - Възможност за записване на проекта в профила на потребителя.
    - Опция за запазване в различни формати (например текстов файл, изображение).
  + Експортиране:
    - Експортиране в PNG и JSON формат.
  + Споделяне:
    - Споделяне на готовата творба в социалната мрежи.

### 

### 2.3 Нефункционални изисквания

* **Изпълнение и ефективност:**
  + Бързо зареждане на страниците и съдържанието
  + Асинхронна обработка на заявки с минимално време за отговор
  + Оптимизирано управление на база данни с бързо извличане на данни
* **Сигурност:**
  + Шифроване на пароли в базата данни
* **Удобство за потребителя:**
  + Интуитивен и адаптивен интерфейс (responsive дизайн)
  + Поддръжка на различни устройства и браузъри
  + Лесна навигация и достъпност
* **Поддръжка и мащабируемост:**
  + Лесно добавяне на нови функционалности
  + Възможност за обработка на голям брой потребители и публикации
  + Добра документация и инструкции за работа

### 2.4 Ползи от реализацията

Реализацията на проекта **CharCraft** предоставя редица предимства както за потребителите, така и за администраторите на платформата:

* **За потребителите:**
  + Възможност за изразяване на креативност чрез уникална форма на изкуство
  + Лесно споделяне на творби с приятели и последователи
  + Изграждане на общност от хора с подобни интереси
  + Мотивация за развитие на артистични умения
* **За пазара:**
  + Запълване на ниша за специфична форма на дигитално изкуство
  + Потенциал за монетизация чрез премиум функции или реклама
  + Възможност за колаборация с дизайнери и артисти

## 3. Теория – анализ и проектиране на решението

### 3.1 Декомпозиция на приложението

Приложението е структурирано по модел на разделяне на отговорностите (Separation of Concerns), като основните му части са:

* **Frontend (публична част на приложението)** – отговаря за потребителския интерфейс и взаимодействието с потребителя.
* **Backend (сървърна част на приложението)** – обработва бизнес логиката, заявки и комуникацията с базата данни.
* **Database (база данни)** – съхранява потребителска информация, рисунки и коментари.

### 3.2 Модули и компоненти

##### 3.2.1 Frontend модули

Frontеnd частта на приложението е разположена в папка public и съдържа следните основни компоненти:

* **HTML файлове:**
  + drawpage.html – страницата за рисуване, съдържаща платно за създаване на ASCII рисунки.
  + feed.html – страницата за преглед на качените рисунки от различни потребители.
  + profile.html – потребителски профил с подробности за създадените рисунки и активност.
  + registration-and-logging-in.html – страница за регистрация и влизане.
  + hero.html – начална страница
* **CSS файлове (в папка css)** – съдържат стиловете за изгледа на приложението.
* **JavaScript файлове (в папка js)** – отговарят за динамичното поведение и взаимодействие с бекенда чрез заявки.

##### 3.2.2 Backend модули

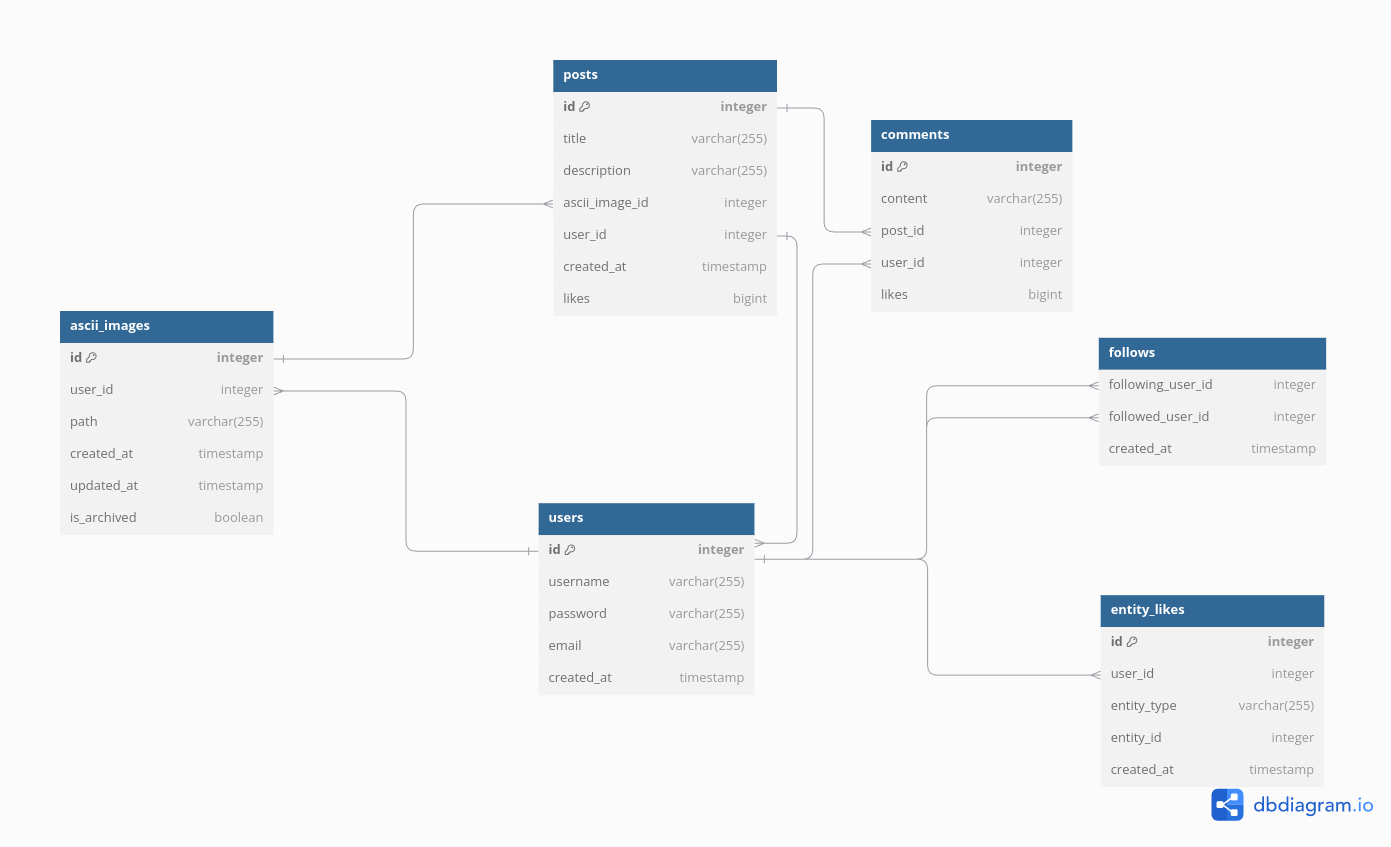
Backend частта на приложението се намира в папка backend и съдържа:

* **Контролери (Controllers):** Обработват HTTP заявките и взаимодействат с моделите и базата данни.
* **Модели (Models):** Представляват структурирането на данните и връзките към базата.
* **Маршрути (Routes):** Определят кои заявки да се насочват към съответните контролери.
* **Конфигурация (config):** Настройки на приложението като връзки към бази данни и API ключове.
* **Съхранение (Storage):** Директория за съхранение на изображения и други файлове.

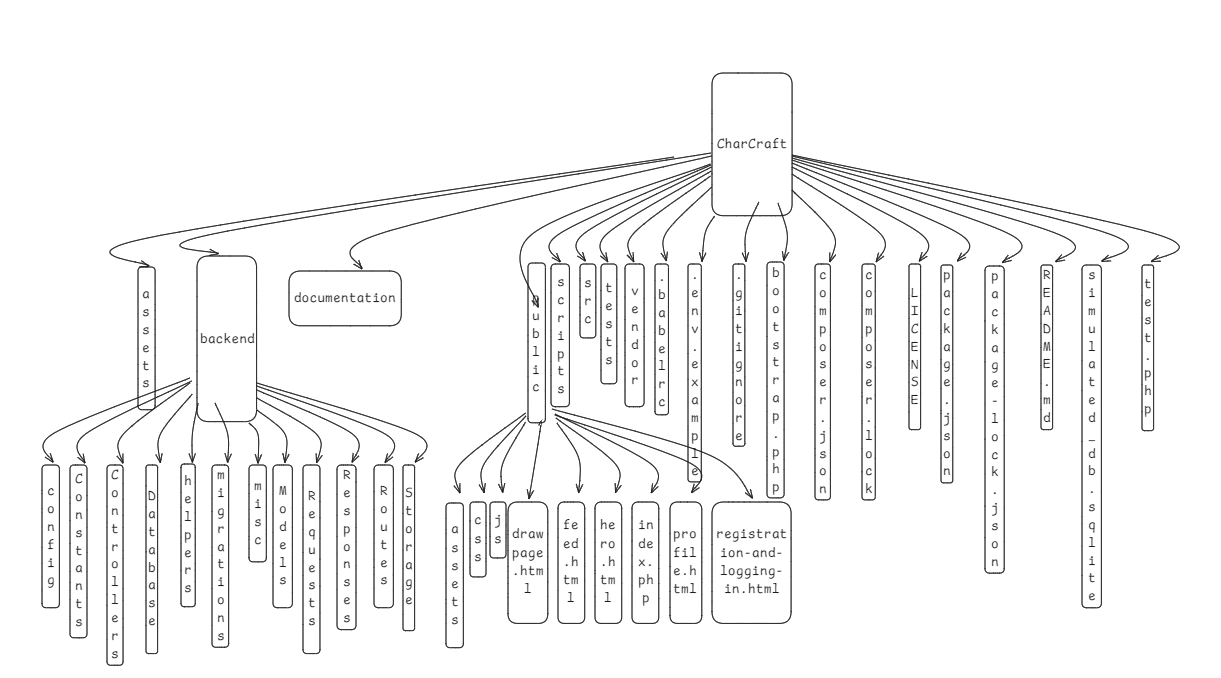
##### 3.2.3 База данни

Приложението използва MySQL база данни. Основните таблици включват:

* **users** – съдържа информация за регистрираните потребители.
* **ascii\_images**– съхранява информация за създадените рисунки.
* **posts** - съхранява информация за създадените постове.
* **comments** – съхранява потребителските коментари към рисунките.
* **entity\_likes** – записва харесванията на потребителите.
* **follows** - съхранява връзките между потребителите, които се следват.



**Основна структура на файловете:**



#### 3.3 Части на приложението

Приложението е разделено на следните основни части:

1. **Регистрация и вход:**
   * Позволява на потребителите да създадат профил и да влязат в системата.
2. **Създаване на рисунки:**
   * Потребителите могат да рисуват с ASCII символи и да запазват своите творби.
3. **Преглед на рисунки:**
   * Възможност за разглеждане на рисунки от други потребители и взаимодействие чрез коментари и харесвания.
4. **Потребителски профил:**
   * Показва информация за създадените рисунки, харесвания и коментари.

#### 3.4 Ползи от избраната архитектура

Избраната архитектура на приложение осигурява следните предимства:

* **Модулност** – лесно разширяване и поддръжка на отделните компоненти.
* **Разделение на отговорностите** – осигурява ясна структура и по-лесна навигация в кода.
* **Гъвкавост** – позволява лесно надграждане на функциите и внедряване на нови модули.
* **Лесна интеграция** – възможност за интеграция с външни услуги и API-та.

С тези характеристики, проектът CharCraft е създаден да бъде ефективен, лесен за употреба и с потенциал за разширяване.

## 4. Използвани технологии

За разработката и работата на приложението **CharCraft** са използвани следните технологии, инструменти и среди:

### 4.1. Основни технологии и програмни езици

* **PHP (версия 8.3)** – използван за сървърната част и обработка на заявките.
* **JavaScript (ES6+)** – за клиентската част, интерактивност и динамична работа с DOM.
* **HTML5** – за създаване на основната структура на уеб страниците.
* **CSS3** – за стилизиране и оформяне на потребителския интерфейс.
* **SQL (MySQL)** – за съхранение на данни и управление на базата данни.
* **Docker**
* **Bash**

### 4.2. Системни среди и инструменти

* **Операционна система:**
  + Windows 10/11 (основна среда за разработка)
  + Linux
  + MacOS
* **Сървърна среда:**
  + Nginx HTTP Server
  + PHP вграден сървър (за локално тестване)
* **База данни:**
  + MySQL (основна база данни за приложението)

### 4.3. Инструменти за разработка

* **Visual Studio Code** – основна среда за разработка с разширения за PHP, JS и HTML.
* **PhpStorm** - основна среда за разработка на PHP
* **Postman** – за тестване на API заявките.
* **Git** – за управление на версиите и сътрудничество по проекта.
* **Composer** – dependency мениджър за PHP зависимости.
* **npm** – за работа с фронтенд зависимости и билд процеси.
* **docker** - за създаване на локална среда за разработка

### 4.4. Изисквания за работа на приложението

За да може приложението да работи коректно, са необходими следните компоненти и конфигурации:

* Инсталиран **PHP 8.x**.
* Стартиран уеб сървър (Apache или PHP built-in server).
* Инсталиран **Composer** за PHP зависимости.
* Инсталиран **Docker**
* Инсталиран **Node.js** и **npm** за управление на JS библиотеки.
* База данни MySQL с предварително изпълнени миграции.

## 5. Инсталация, настройки и DevOps

### Необходими инструменти и технологии:

* Docker (последна версия)
* Docker Compose
* Браузър за достъп до приложението

### Стъпки за инсталация:

#### 1. Клониране на хранилището:

Уверете се, че имате инсталиран Git и изпълнете следната команда:

* git clone <URL на хранилището>

cd <име на хранилището>

#### 2. Настройки на конфигурацията:

Редактирайте файла .env, който съдържа следните променливи:

DB\_HOST=mysql

DB\_USERNAME=admin

DB\_PASSWORD=admin

DB\_DATABASE=char\_craft

Уверете се, че сте задали подходящи стойности за променливите на базата данни.

#### 3. Създаване и стартиране на контейнерите:

Стартирайте Docker Compose, за да създадете и стартирате необходимите услуги:

docker-compose up --build -d

Тази команда ще изгради и стартира контейнерите за приложението и базата данни.

#### 4. Достъп до приложението:

След успешното стартиране, приложението ще бъде достъпно на адрес: http://localhost:3000

### Примерни настройки за база данни:

В Docker Compose конфигурацията базата данни е дефинирана по следния начин:

version: "3.8"

services:

char\_craft:

image: shinsenter/php:8.4-fpm-nginx

environment:

APP\_PATH: /var/www/html

DOCUMENT\_ROOT: /public

container\_name: char\_craft

ports:

- "3000:80"

restart: unless-stopped

volumes:

- ./:/var/www/html

- ./scripts/entrypoint.sh:/startup/00-migration

working\_dir: /var/www/html

networks:

- app\_network

depends\_on:

- mysql

mysql:

image: mysql:8.0

container\_name: mysql\_db

restart: unless-stopped

ports:

- "3306:3306"

environment:

MYSQL\_DATABASE: char\_craft

MYSQL\_ALLOW\_EMPTY\_PASSWORD: "yes"

MYSQL\_USER: admin

MYSQL\_PASSWORD: admin

volumes:

- mysql\_data2:/var/lib/mysql

networks:

- app\_network

networks:

app\_network:

volumes:

mysql\_data2:

### Примерни акаунти за вход:

* **Обикновен потребител:**
  + Потребителско име: admin
  + Парола: admin123

## 6. Кратко ръководство на потребителя

## Основи:

Всички страници съдържат следното:  
**Главна навигация**:

В горната част на страницата има лого, което води до основната страница, както и линкове към секциите Home, Draw, Feed

**Долна част на страницата**:

В подножието на страницата се намира информация за авторските права ("2025 CharCraft. All rights reserved.").

**Основни страници:**

**Hero страница:**

Под приветствието има бутон "Register Now", който води към страницата за регистрация и вход.

**Страница за регистрация и логване:**

**Регистрация:**  
Потребителите, които искат да се регистрират, трябва да попълнят полетата за потребителско име, имейл, парола и потвърждение на паролата. След това трябва да натиснат бутона "Sign up".

**Вход:**  
За вече регистрирани потребители има форма за вход, в която трябва да въведат имейл и парола, след което натискат бутона "Sign in".

**Страница с постове:**

**Създаване на публикация:**

* В първата част на страницата има форма за създаване на публикация, която изисква:
  + Заглавие на публикацията.
  + Описание на публикацията.
  + Изображение във формат ASCII (потребителят трябва да качи файл).
* Когато всички полета са попълнени, потребителят може да натисне бутона "Post" за да публикува съдържанието.

**Отзиви и коментари:**

* Под всяка публикация има лайк бутон, който позволява на потребителите да харесват публикациите.
* Потребителите могат да добавят коментари под публикацията, като попълнят формата за коментари и натиснат "Comment". Новите коментари ще се показват под основната публикация.

**Преглед на публикации:**

* Всички публикувани постове се показват в контейнер, като всяка публикация съдържа информация за потребителя, дата на създаване, изображение, описание, лайкове и коментари.

**Страницата Профил:**

**Потребителска информация:**

* В началото на страницата ще видите заглавие с потребителското име. Под него има бутон "Change Password" за промяна на паролата (функцията е все още неактивна).

**Публикации на потребителя:**

* Под потребителската информация се показват публикациите на потребителя, всяка с малко изображение.
* При клик върху изображението, се показва поп-ъп прозорец с пълната публикация (описание, изображение и други детайли).

**Затваряне на поп-ъп прозорец:**

* За да затворите поп-ъп прозореца, използвайте бутона "x" в горния ъгъл на прозореца.

## Дъска: Функционалности на бутоните

### Цветова палитра

## Цветни бутони: При натискане на бутон от палитрата се задава текущият цвят на четката. Избраният бутон се маркира визуално.

### Изчистване

## Бутони за изчистване: При натискане се показва съобщение за потвърждение. Ако потребителят потвърди, дъската се изчиства напълно.

### Инструменти

## Бутони за инструменти:

## Инструмент "Четка" задава рисуване с четка.

## Инструмент "Кофа" запълва областта около кликнатата точка с избраната функция на четката. (виж по-долу)

### Форми за рисуване

## Бутони за форма:

## "Кръг" задава четка с форма на кръг.

## "Квадрат" задава четка с форма на квадрат.

### Видове четки

## Бутони за четки:

## "Normal" за стандартно рисуване.

## "Color Only" за оцветяване без промяна на символи.

## "Character Only" за поставяне на символи без промяна на цвят.

## "Embolden" за удебеляване на символи.

## "Fade" за избледняване на символи.

## "Erase" за изтриване.

### Цветови избор

## Цветови избор:

## Клик на бутона отваря цветова палитра за избор. След избор, цветът се задава като текущ.

## Zoom и местене на дъската

## Местене: Задръжте десния бутон на мишката и плъзнете, за да преместите дъската.

## Zoom:

## Завъртете скрола на мишката, за да увеличите или намалите мащаба.

## Мащабирането е центрирано върху позицията на мишката.

## Центриране: Дъската автоматично се центрира в прозореца при стартиране.

### Рисуване

## Ляв бутон: Рисуване на дъската.

## Функционалност: Зареждане на изображение и преобразуване в ASCII

## С помощта на бутона "Load Image" потребителят може да зареди изображение и да го преобразува в ASCII арт. След натискане на бутона, се отваря прозорец за качване на изображение, в който могат да бъдат направени настройки за преобразуване. Резултатът е визуално ASCII представяне на изображението.

## Как да заредите изображение

## Натиснете бутона "Load Image".

## Ще се отвори прозорец за качване на файл.

## Изберете изображение от устройството си.

## След качване, изображението автоматично ще бъде обработено спрямо текущите настройки и ще бъде визуализирано за предварителен преглед.

## Настройте опциите за преобразуване (по избор):

## Резолюция.

## Яркост и контраст.

## Откриване на ръбове.

## Набор от ASCII символи.

## Прегледайте резултата.

## Изображението ще бъде преобразувано в ASCII дъска.

## Настройки за обработка на изображението

### Резолюция

## Цел: Намаляване на размера на изображението за по-лесно преобразуване.

## Опции:

## Resolution X: Максимален брой символи по хоризонтала.

## Resolution Y: Максимален брой символи по вертикала.

## Автоматично скалиране: Съотношението на страните се запазва.

### Яркост и контраст

## Яркост: Позволява увеличаване или намаляване на яркостта.

## Гама корекция: Подобрява контраста на изображението.

### Откриване на ръбове

## Цел: Подчертаване на контурите на изображението.

## Опции:

## Edge Detection: Включване/изключване на откриването на ръбове.

## Edge Detection Threshold: Настройка на чувствителността за откриване на ръбове (стойност от 0 до 255).

### Набор от ASCII символи

## Пълен набор: Използва широк спектър от ASCII символи за по-детайлно изображение.

## Редуциран набор: Използва ограничен брой символи за по-опростено изображение.

## Съвети:

## Уверете се, че сте запазили работата си, преди да използвате бутона за изчистване или да ресетирате страницата.

## 

## 7. Примерни данни

## Тестване на js код: Windows:

## Ако npm вече е инсталиран, можете да пропуснете тези стъпки.

#### Стъпка 1: Инсталиране на Chocolatey

## Отворете PowerShell като Администратор и изпълнете следната команда: powershell -c "irm https://community.chocolatey.org/install.ps1 | iex"

#### Стъпка 2: Инсталиране на Node.js

## С помощта на Chocolatey, инсталирайте Node.js: choco install nodejs-lts --version="22.11.0"

#### Стъпка 3: Проверка на инсталацията

## След инсталирането, може да е необходимо да рестартирате средата, преди да извършите проверката. След това изпълнете следните команди:

## node -v # Очакван резултат: v22.12.0 npm -v # Очакван резултат: 10.9.0 Тестване:

### Стъпка 1: Инсталиране на зависимости

## Отидете до основната директория на проекта и изпълнете: npm install

### Стъпка 2: Стартиране на тестове

## За да изпълните тестовете, просто изпълнете: npm test

## Отстраняване на проблеми:

## Ако зависимостите не се инсталират с командата npm install, можете да ги инсталирате индивидуално по следния начин:

## npm install --save-dev jest npm install --save-dev jest-environment-jsdom npm install --save-dev @babel/core @babel/preset-env babel-jest npm install --save-dev jest-canvas-mock

## 

## 

## 8. Описание на програмния код Дъската за рисуване:

### 1. Клас Character

* Представлява символ с цвят.
* Свойства:
  + character: Символът.
  + color: Цветът на символа.
* Метод:
  + displayDetails(): Извежда символа и цвета в конзолата.

### 2. Клас CharacterBoard

* Управлява двумерна матрица от символи (boardMatrix).
* Свойства:
  + rows, cols: Размерите на дъската.
  + defaultCharacter, defaultColor: Стойности по подразбиране за символи и цветове.
  + boardMatrix: Съдържа обекти Character.
* Методи:
  + resetBoard(), clearBoard(): Изчистват или ресетират дъската.
  + isPositionValid(row, col): Проверява дали дадена позиция е валидна.
  + fillBoardWithCharacter(character, color): Запълва дъската с даден символ и цвят.
  + colorCell(row, col, character, color): Променя символа или цвета на определена клетка.
  + emboldCell(), fadeCell(): Подсилват или избледняват символа в клетката.
  + exportBoardAsJSON()/importBoardFromJSON(boardData): Експорт/импорт на дъската в/от JSON формат.
  + setBoardSize(rows, cols): Променя размера на дъската.

### 3. Клас CanvasBoard

* Разширение на CharacterBoard, добавящо визуализация с HTML <canvas>.
* Свойства:
  + canvas, context: HTML елементи за рисуване.
  + cellWidth, cellHeight: Размер на клетките.
* Методи:
  + initialiseContainer(container): Инициализира контейнера за канваса.
  + redrawBoard(), \_setCell(): Рисуват или актуализират клетките на екрана.

### 4. Клас Brush

* Управлява инструмент за рисуване.
* Свойства:
  + mouseRadius: Радиус на действие.
  + drawCharacter, drawColor: Символ и цвят за рисуване.
  + brushType: Тип на четката (нормална, гумичка, само символи и т.н.).
* Методи:
  + drawOnPosition(drawingBoard, row, col): Рисува в определена позиция.
  + bucketOnPosition(drawingBoard, row, col): Излива цвят на област с еднакви символи и цветове.
  + drawLine(drawingBoard, fromX, fromY, toX, toY): Рисува линия между две точки.

### 5. Клас DrawingBoardUI

* Обединява взаимодействието на потребителя с CanvasBoard и Brush.
* Свойства:
  + drawingBoard, brush: Рисувателната дъска и четката.
  + isLeftMouseDown, isRightMouseDown: Флагове за състоянието на мишката.
* Методи:
  + init(): Инициализира потребителския интерфейс.
  + enableBoardUI()/disableBoardUI(): Активира/деактивира взаимодействието.
  + getClickedCellCoordinates(event): Изчислява клетката, върху която е кликнато.
  + leftMouseDown(event), rightMouseDown(event): Обработват натискания на левия/десния бутон.
  + draw(event), leftMouseMove(event): Управляват процеса на рисуване.
  + Управлява взаимодействието на бутоните с четката.

### 6. Други Константи и Типове

* **BrushShape**: Определя формата на четката (кръгла или квадратна).
* **ToolType**: Определя типа инструмент (четка или кофа за запълване).
* **BrushType**: Задава специфични поведения на четката.

**7. Клас ImageConverter**Класът ImageConverter предоставя методи за преобразуване на изображения в ASCII изкуство. Той позволява обработка на изображения чрез няколко стъпки: корекция на яркостта, гамма корекция и откритие на ръбове. Основната функция на класа е да преобразува пикселите на изображението в символи, които се визуализират като ASCII изображения.

### Основни функции:

1. **parseImageToBoard**: Основен метод, който приема изображение и опции, извлича пикселите и ги обработва, след което създава "борд" (матрица) със символи и цветове.
2. **createBoardFromPixels**: Създава и попълва борда със символи и цветове от обработените пиксели.
3. **applyBrightnessFactor**: Променя яркостта на пикселите.
4. **applyGammaCorrection**: Прилага корекция на гамата на изображението.
5. **applyEdgeDetection**: Прилага детекция на ръбове, като използва операциите на Собел.
6. **getImagePixels**: Извлича пикселите на изображението, като ги мащабира спрямо зададените резолюции.
7. **getPixelSymbol**: Изчислява съответния ASCII символ за даден пиксел в зависимост от средния цвят на пиксела.
8. **getPixelColorHex**: Връща хекс кода на цвета на пиксела.

### Опции за конфигурация:

* Можете да зададете поредица от опции като яркост, корекция на гамата, и детекция на ръбове, чрез класа ImageParseOptions, който предоставя пълна персонализация на обработката на изображението

## Frontend

### 1. drawingboard.js

Отговаря за основната функционалност на приложението – рисуване върху платно с ASCII символи.

**Основни функции и класове:**

* DrawingBoardUI: Управлява потребителския интерфейс на рисувалното платно.
* CanvasBoard: Клас, който представлява платното за рисуване и обработва пикселите като ASCII символи.
* Brush: Инструмент за рисуване с различни настройки като форма, размер и вид на четката.
* ImageConverter: Клас за конвертиране на изображения в ASCII символи, като предоставя различни опции за персонализиране (яркост, резолюция, гама корекция).
* **Основни DOM елементи:**
  + Бутон за зареждане на изображение (image-loader-btn).
  + Входни полета за настройка на резолюция, яркост и други параметри.
  + Бутон за изтегляне на готовото изображение в PNG формат.

**Основни функции в скрипта:**

* getImageOptions(): Събира потребителските настройки за обработка на изображението.
* loadPreviewBoard(): Зарежда предварителен изглед на ASCII изображението.
* showPopup() и hidePopup(): Контролират показването и скриването на диалоговия прозорец за настройки.
* document.addEventListener('DOMContentLoaded', ...): Инициализира основните функционалности при зареждане на страницата.

### 2. profile.js

Отговаря за управлението и визуализирането на потребителските профили.

**Основни функции:**

* generateProfile(profileData): Генерира HTML елементи за показване на потребителския профил, включително публикации и информация за потребителя.
* Зареждане на потребителските данни чрез JSON структура и визуализиране на постове с изображения и коментари.
* Възможност за отваряне на публикации в отделен прозорец и затваряне с бутон.

**Основни елементи:**

* Име на потребителя.
* Бутон за промяна на паролата.
* Секция с изображения на създадените ASCII рисунки.
* Коментари и харесвания за всяка публикация.

### 3. registration-and-logging-in.js

Отговаря за регистрацията и влизането в системата.

**Основни функции:**

* submit събития за формите за регистрация и вход.
* Изпращане на данни към бекенда чрез fetch() заявка към сървъра.
* Обработка на успешна или неуспешна регистрация чрез alert съобщения.
* Пренасочване към потребителския профил при успешен вход.

**Основни действия:**

* Изпращане на регистрационна форма към /register с метод POST.
* Изпращане на логин форма към /login с метод POST.
* Валидация и обработка на грешки при неуспешни заявки.

### 4. feed.js

Отговаря за зареждане и визуализиране на постове в главния поток от съдържание.

**Основни функции:**

* generatePost(post): Създава HTML елементи за публикация, включително заглавие, изображение, харесвания и коментари.
* Добавяне на коментари към публикации чрез форма.
* Функционалност за харесване на публикации, визуализиране на броя харесвания.
* Използване на fetch() за зареждане на постове от бекенд API.

**Основни елементи:**

* Секция с публикации с текстово и визуално съдържание.
* Форма за добавяне на коментари.
* Икона за харесване с динамична промяна.

## 9. Приноси на студента, ограничения и възможности за бъдещо разширение Александър Македонски:

## Дъската за рисуване

## Функционалностите около UI-я на дъската - с допълнителна разработка от Памела Славчева.

## Конвертора на изображения

## Тестове на функционалността на класовете. Бъдещо развитие: Желая да имплементирам webgl за забързване на rendering-а на изображенията, за по-добра работа с приложението.

## Памела Славчева:

1. Всички html и css файлове.
2. Целия JavaScript код за изпращане на заявки към backend и тяхната обработка.
3. Динамично създаване на постове, потребителски профил и фийд. Обработка на коментари.
4. Функционалностите около UI-я на дъската - избиране на цвят, избиране на буква, свързване на бутоните към функционалността на дъската, написана от Александър Македонски.

Бъдещо развитие:

* Подобряване на интерактивността на съществуващите страници, чрез клавишни комбинации.
* Добавяне на draft рисунки, по които може да се работи с прекъсване и не са публикувани или свалени от потребителя.
* Търсене във фийда и възможност за създаване на групи.
* “Споделени” дъски, на които могат да рисуват няколко потребителя едновременно.

## Васил Христов:

1. Всеки един php файл
2. Изграждане на Model View Controller framework, с който по-лесно да настъпи разработката
3. Динамично четене на environment променливи
4. Функционалност за логване на грешки на сървъра
5. Динамични конфигурации
6. Псевдо ORM, с който да не се пише чист SQL
7. Фасада за работа с файлове
8. Фасада за автентикация и оторизация
9. Динамични заявки, валидации на полета в заявки, валидация на бисквитки в заявките
10. Динамични отговори на заявките
11. Маршрутизатор за заявките
12. Функционалност за миграции на сървъра
13. Докеризиране на приложението
14. Настройка на хедъри и бисквитки
15. Работа с константи в php рамката

Бъдещо развитие:

* да работи

## 10. Какво научих

## Александър Македонски:

Основно:

1. Обработка на real-time събития в браузър;
2. Изграждане на динамичен потребителски интерфейс в браузър;
3. Преобразуване на изображения в ASCII арт;
4. Алгоритми като Собел за откриване на ръбове в изображения;
5. Как да прилагам корекции на яркостта и гамата върху изображение;
6. Работа с няколко различни резолюции едновременно, за изобразяване на подходящи размери за нуждите на потребителите;
7. Обектно-ориентирано програмиране в JS;
8. Манипулиране с DOM;
9. Интеракция с графични обекти;
10. Работа с масиви и обекти в JS.

Допълнително (покрай колегите):

1. Изпращане на заявки към сървър чрез JS, свързване на endpoint-и;
2. Обработка на отговори от заявки;
3. Обработка и валидация на данни чрез php на сървър;
4. Автентикация.

## Памела Славчева:

## Уеб разработка (Frontend & Backend)

* Научих се да работя с **HTML, CSS и JavaScript**, за да изградя потребителски интерфейс, който е интуитивен и функционален.
* Разбрах важността на **DOM манипулацията**, събитията и начините за работа с потребителския вход.
* Използвах **Fetch API** за комуникация с бекенд сървъра и работа с **JSON** данни.
* Запознах се с **основите на бекенд разработката**.

## Проектиране на потребителско изживяване (UX/UI)

* Научих как да проектирам **интуитивен и достъпен интерфейс**, който улеснява потребителите при създаване и споделяне на съдържание.
* Разбрах важността на **добрата организация на елементите**, като използване на модални прозорци, лесен достъп до инструменти и удобна навигация.
* Тестване на различни UI решения, за да се подобри удобството на платформата.

## Работа в екип и управление на проекти

* Научих как да планирам и изпълнявам **стъпка по стъпка разработката на един проект**, като разделям задачите на малки подзадачи.
* Разбрах важността на **гъвкавата методология (Agile/Scrum)** при работа върху по-големи проекти.
* Запознах се с инструменти за **управление на код**, като Git и GitHub, за сътрудничество и проследяване на промените.

## Дебъгване и тестване на приложението

* Научих как да използвам **инструменти за дебъгване** в браузъра (Chrome DevTools) за анализ на грешки и оптимизация на кода.
* Разбрах важността на тестването на приложението върху различни устройства и браузъри за осигуряване на съвместимост.
* Използване на логове и съобщения за грешки за откриване и коригиране на проблеми в кода.

## Васил Христов:

* Разбрах, че трябва първо да настроиш сървъра, после да напишеш кода.
* Разбрах, че кода трябва да се тества докато се пише, защото, ако се остави да се свързва всичко в последния момент -> няма да работи нищо
* Работа с JWT tokens и бисквитки
* Как да се изградят файловите директории на един монолитен проект. Настройване на nginx server с php-fpm
* Работа с примитиви като PDO
* Auto resolving в PHP
* Написах си цял фреймуърк, за да не пиша на vanilla php

## 11. Използвани източници

**Александър Македонски:**

Мозъка на Александър Македонски, последно посетен 28 декември, 2024.

**Памела Славчева:**

1. **Duckett, J.** (2014). *HTML and CSS: Design and Build Websites.* John Wiley & Sons.
2. **Freeman, E., Robson, E., & Sierra, K.** (2021). *Head First HTML and CSS.* O'Reilly Media.
3. **Resig, J., Bibeault, B., & Maras, Y.** (2021). *Secrets of the JavaScript Ninja, 2nd Edition.* Manning Publications.
4. W3Schools. (n.d.). *JavaScript Fetch API.*
   * https://www.w3schools.com/js/js\_api\_fetch.asp
   * Дата на посещение: 27.01.2025
5. CSS-Tricks. (n.d.). *A Complete Guide to Flexbox.*
   * https://css-tricks.com/snippets/css/a-guide-to-flexbox/
   * Дата на посещение: 27.01.2025

**Васил Христов:**

Любезното съдействие на ChatGPT в материята на фронтенда и 5г опит в сферата. Както и:

1. Shinsenter/php: <https://github.com/shinsenter/php/tree/main>

Предал (подпис): ………………………….

/*1MI0800046, Памела Славчева , КН, 2*/

Приел (подпис): ………………………….

/проф. д-р *Милен Петров*/