

# Documentació tècnica de



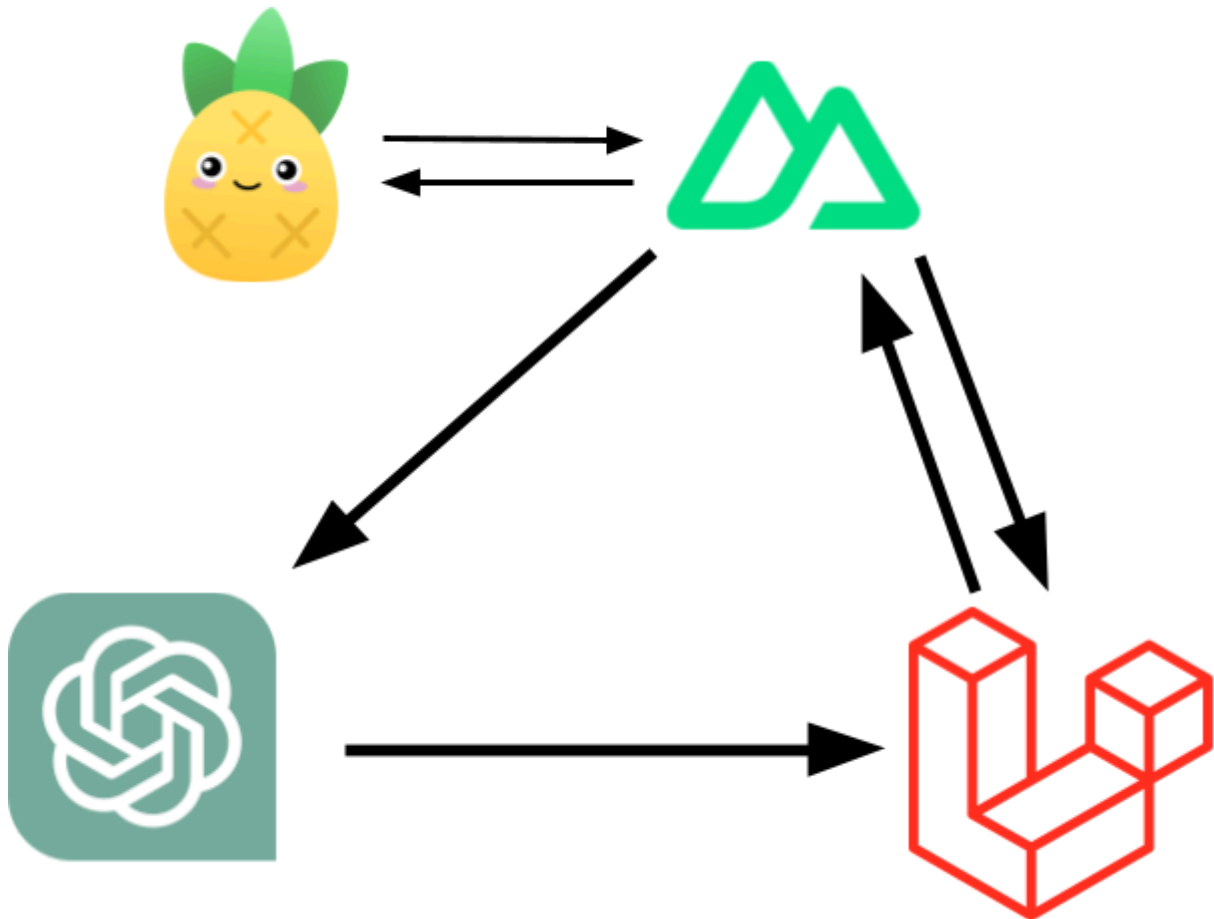
Alex Delgado  
Miquel Barcelo  
Eric Rodriguez  
Aitor Barreiro

<b>1. Arquitectura de l'aplicació.....</b>	<b>3</b>
<b>2. Rutes de l'aplicació:.....</b>	<b>3</b>
○ Getters:.....	4
○ Post:.....	4
○ Put:.....	4
○ Delete:.....	4
○ Api Chat GPT:.....	5
<b>3. Esquema de base de dades.....</b>	<b>6</b>
○ Imatge del phpMyAdmin de l'esquema amb les relacions de la base de dades:.....	6
○ Explicació de l'esquema de la base de dades:.....	6
<b>4. Esquema de components frontend.....</b>	<b>7</b>
○ Imatge del esquema del nostre front:.....	7
○ Explicació del esquema:.....	7
<b>5. Documentació de codi frontend:.....</b>	<b>8</b>
○ Communication Manager:.....	8
○ Registre.....	11
○ Chat Rutina o Dieta:.....	16
<b>6. Documentació de codi backend.....</b>	<b>20</b>
○ Laravel:.....	20
<b>7. Disseny.....</b>	<b>24</b>
Imatge corporativa.....	24
Tipografia.....	25
Icones.....	25
Pàgines.....	26
CSS del header:.....	27
CSS de la barra de navegació:.....	30
Les pantalles principals de la nostra aplicació són:.....	32
Exemples de codi css:.....	36
<b>8. Desplegament.....</b>	<b>55</b>
Docker:.....	55

## 1.Arquitectura de l'aplicació

La nostra aplicació s'utilitzen les següents arquitectures:

- Nuxt
- Laravel
- Api de ChatGpt
- Docker



## 2.Rutes de l'aplicació:

La nostra aplicació totes les rutes les tenim un fitxer javascript que es diu communication Manger en el que tenim tots els fetch alla.

L'arxiu està dividit pels diferents tipus de fetch, els GET, POST, PUT, DELETE i els que es fan a la API de Chat GPT.

## ○ Getters:

- **getDatosUsuario:** `fetch(`${url}/usuari/${idUsuario}`)`:  
Aquesta ruta serveix per rebre del back totes les dades de 1 usuari concret.
- **getTotsUsuaris:** `fetch(`${url}/tots-usuaris`)`:  
Aquesta ruta serveix per rebre del back tots els usuaris que estiguin en el back
- **getUsuariosChat:** `fetch(`${url}/chatUsuaris/${idUsuario}`)`:  
Amb aquesta ruta reps els usuaris que son amics.
- **getDatosEjercicio:** `fetch(`${url}/ejercicios`)`:  
Amb aquesta ruta reps totes les dades dels exercicis que estan en la base de dades.
- **getDatosAlimentos:** `fetch(`${url}/alimentos`)`:  
Amb aquesta reps totes les dades del aliments que están en la base de dades.
- **getDieta(idUsuario):** `fetch(`${url}/dieta/${idUsuario}`)`:  
Rebs la dieta d'un sol usuari de la base de dades.
- **mostrarUltimoMensajeEllos(idUsuario, idAmigo):**  
`fetch(`${url}/ultim-missatge/${idUsuario}/${idAmigo}`)`:  
reps les dades dels últims missatges del chat entre els amics.

## ○ Post:

- **iniciarSesion(email, contrasenya)** `fetch(`${url}/loguejat`)`:  
enviem les dades de l'inici de sessió al back
- **enviarRutinaAlServidor(rutina)** `fetch(`${url}/guardar-rutina`)`:  
un cop generada la rutina la guardem al back
- **enviarDietaAlServidor(rutina):** `fetch(`${url}/guardar-dieta`)`:  
Es lo mateix que rutina pero amb dietes. Guardar la informació al back
- **enviarSolicitudAmistad(usuarioEnviald, usuarioRecibeld)**  
`fetch(`${url}/enviar-solicitud`)`:  
serveix per enviar la sol·licitud al back per guardar a qui li has enviat per després saber qui es amic teu.

## ○ Put:

- **actualizarDatosUsuario(idUsuario, formData):**  
`fetch(`${url}/editar-usuari/${idUsuario}`)`:  
server per els camps que actualitzis s'actualitzin al back també.

## ○ Delete:

- **borrarRutinaDia(idUsuario)** `fetch(`${url}/eliminar-rutina-hoy/${idUsuario}`)`  
si hi ha una rutina creada avui la borra de la base de dades.
- **borrarDietaHoy(idUsuario):** `fetch(`${url}/eliminar-dieta-hoy/${idUsuario}`)`:  
Si hi ha dieta d'avui la borra de la base de dades.
- **borrarUsuario(idUsuario):** `fetch(`${url}/eliminar-usuari/${idUsuario}`)`  
els administradors poden esborrar els usuaris.
- **deleteRutinaByDate(idUsuario, fecha):** `fetch(`${url}/rutinas/${idUsuario}/${fecha}`)`:  
Quan l'usuari consulta rutines antigues pot esborrar-les.

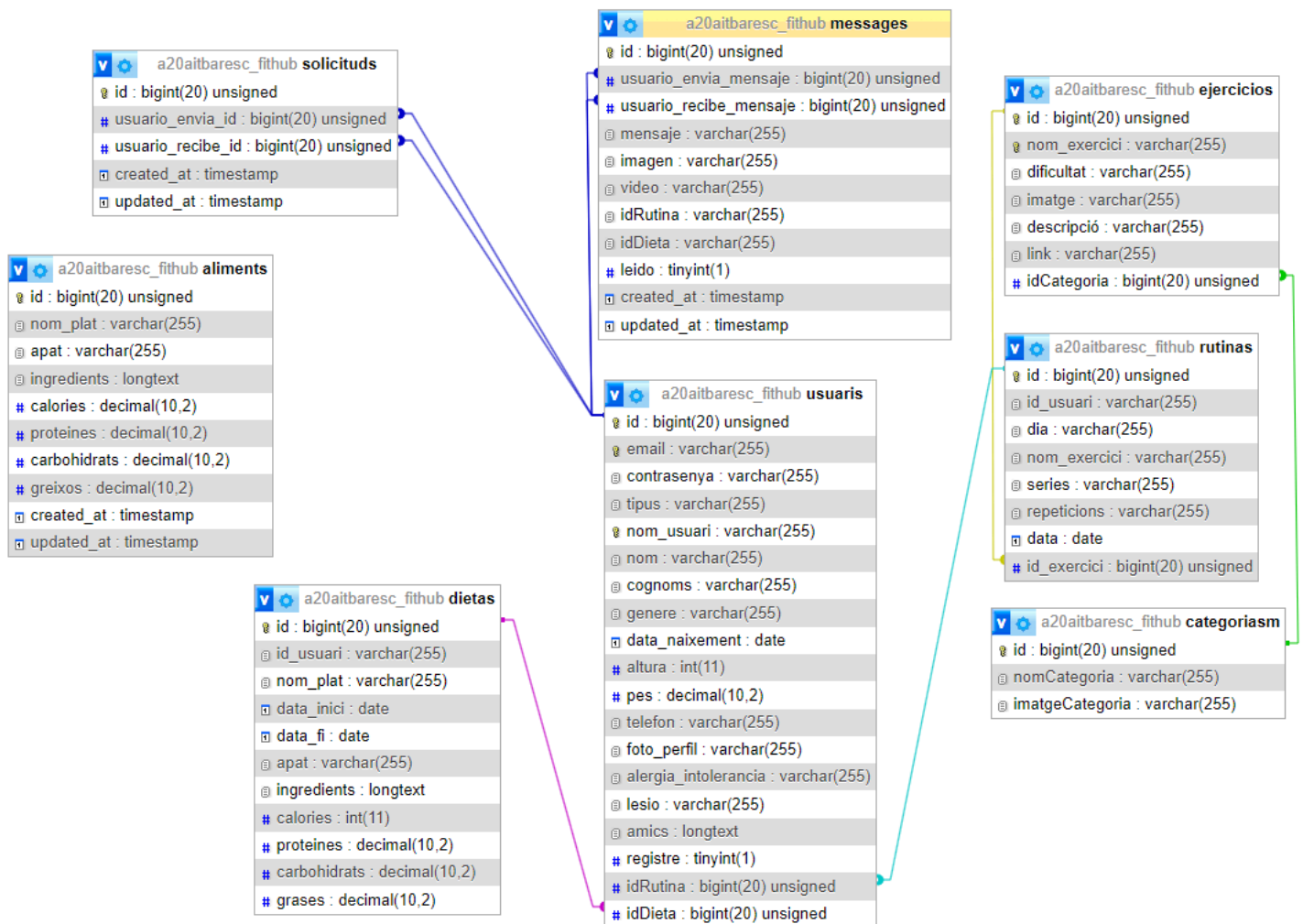
- **deleteDietaByDate(idUsuario, fecha):** fetch(`\${url}/dietas/\${idUsuario}/\${fecha}`): Serveix igual que rutina pero en Dieta, esborra les dietes d'aquella data.

## ○ **Api Chat GPT:**

- **enviarMensajeOpenAIRutina(message, ejercicios, datosUsuario):**  
Aquesta funció/ruta, agafa el missatge de l'usuari, l'exercici i les dades de l'usuari per generar una rutina per a l'usuari.
- **enviarMensajeOpenAIDieta(message, alimentos, datosUsuario):**  
Es igual que la de rutina pero amb dieta, tenen props diferents i llavors cada una fa una funció es diferent pero la base es la mateixa.
- **enviarMensajeOpenAI(message):**  
Aquesta es la ruta sencilla ja que lo que fa es un assistent el prop es més senzill i només agafa el missatge de l'usuari i respon lo que li pregunten.

### 3. Esquema de base de dades

- Imatge del phpMyAdmin de l'esquema amb les relacions de la base de dades:



- Explicació de l'esquema de la base de dades:

La nostra aplicació gira tot entorn a l'usuari

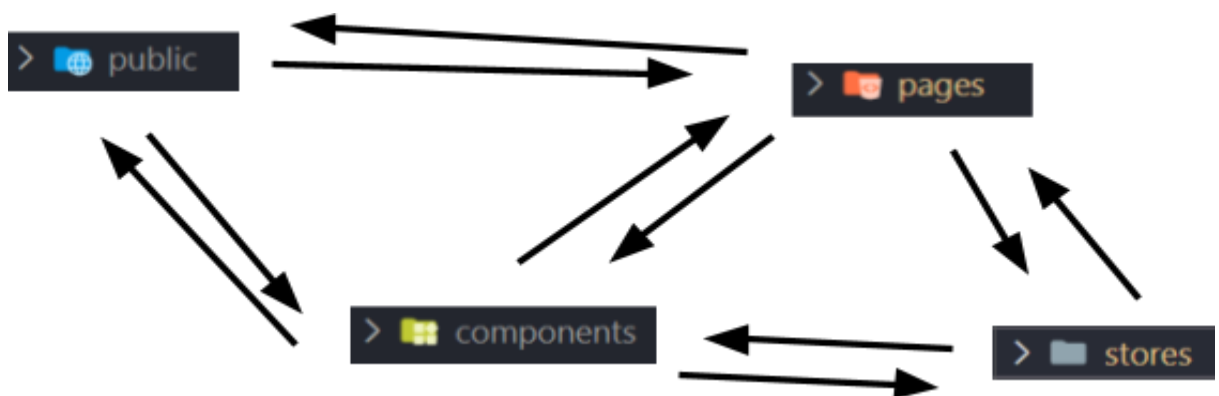
- L'usuari pot tenir més d'una rutina i una rutina esta associada a molts usuaris, les rutines es vinculen amb l'usuari a traves de l'id d'usuari.
- Un usuari pot tenir més d'una dieta i una dieta pot estar associada a molts usuaris, per saber o unir dietes amb usuaris es vinculen amb l'id d'usuari.
- Un usuari pot enviar moltes sol·licituds i les sol·licituds poden ser de molts usuaris. Amb les sol·licituds ls vinculem amb l'id d'usuari i també agafem l'id de l'altre usuari per que la pugui rebre.
- Un usuari pot enviar molts missatges i un missatge pot ser enviat per molts usuaris. Amb l'id es vincula l'usuari que envia aquell missatge també l'id de a qui va destinat

per poder-ho enviar i així aconseguim també guardar els missatges per quan entris al chat tenir un historial del chat antic.

- Amb rutines es creen a través dels nostres exercicis, llavors amb l'id del exercici vinculem rutina amb exercicis ja que 1 rutina pot tenir molts exercicis i molts exercicis poden ser a moltes rutines.
- Dins d'exercicis tenim categories, 1 exercici pot estar a 1 categoria i 1 categoria pot tenir molts exercicis. Llavors amb l'id de categoria enllacem les diferents taules per tenir un millor control de categories i poder relacionar cada exercici a la seva categoria corresponent.
- No vam enllaçar les taules dieta amb aliments pero fan una funció similar que rutina ja que la ia agafa els aliments de la nostra base de dades per generar dietes. Llavors els nom\_plat son iguals tant a dieta com a aliments, ja que les dietes es creen arrel de la taula d'aliments.

## 4. Esquema de components frontend

- Imatge del esquema del nostre front:



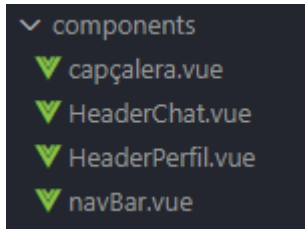
- Explicació del esquema:

- **Pages:**

En pages estan totes les pàgines del nostre projecte. Les pages utilitza la carpeta public que es on agafa les imatges per mostrar rutina o els botons o qualsevol imatge de l'aplicació. També utilitza stores que es on estan tots els fetchs i el pinia. I també utilitza components que es on estan els components com la navbar o la capçalera que s'utilitzen a la majoria de pàgines.

- **Components:**

Conte tots els components que s'utilitzen en les pàgines de pagès. I també s'enllaça amb stores ja que utilitza fetchs i també el pinia per mostrar el nom d'usuari o mostrar la imatge d'usuari.



- capçalera: és el header que apareix en totes les pàgines. conté la foto de perfil amb link al perfil de l'usuari. Al costat de la foto es mostra el nom d'usuari.
- HeaderPerfil: és el header de l'apartat de chat. La principal diferència és que aquest té una lògica implementada per mostrar sollicituds d'amics i afegir amics.
- HeaderChat: és el header que apareix en la pàgina de modificar perfil. La diferència és que aquest té una lògica implementada per modificar la foto de perfil.
- navBar: és la barra de navegació que es mostra a la part inferior de la pàgina. Conté els enllaços a les funcionalitats de l'aplicació (dieta, rutina, home, chat, assessorament).

#### ■ Stores:

Conte dos arxius, el pinia que es el index.js i el communicationManager.js que conté tots els fetch de l'aplicació les urls i les api keys del chat GPT. I es vincula amb pages i components que són els únics que utilitzen el pinia o els fetch o ambdues.

#### ■ Public:

Conte les imatges, logo, icones que s'utilitzen en les pàgines de pages, per això s'enllaça amb pages que es l'únic que utilitza el públic i components.

## 5. Documentació de codi frontend:

### ○ Communication Manager:

El nostre communication Manager estem bastant contents amb ell ja que es un fitxer amb casi tots els fetch ordenats pel tipus de fetch, GETS, POST, PUT, DELETE i API CHAT.

Mostra del nostre communication Manager:



```

1 // const url = 'http://localhost:8000/api';
2 const url = 'http://fithub.daw.inspedralbes.cat/back/public/api';
3 const apiUrl = 'https://api.openai.com/v1/chat/completions';
4 const apiKey = 'sk-Xqz2dcsp7N88gatVPY61T3B1bkFJD8Rmqj86cxTBhqQ6dFhX'; // Usa la variable de entorno
5
6
7 //ejemplo de petición fetch get
8
9 export function getDatosUsuario(idUsuario) {
10   return new Promise((resolve, reject) => {
11     fetch(`${url}/usuari/${idUsuario}`)
12       .then(response => {
13         if (response.ok) {
14           return response.json();
15         } else {
16           reject('Error al obtener los datos del usuario: ' + response.statusText);
17         }
18       })
19       .then(data => {
20         resolve(data);
21       })
22       .catch(error => {
23         reject('Error de red al obtener los datos del usuario: ' + error.message);
24       });
25   });
26 }
27
28 export async function getDatosUsuario2(idUsuario) {
29   try {
30     const response = await fetch(`${url}/usuari/${idUsuario}`);
31     if (!response.ok) {
32       throw new Error('Error al obtener los datos del usuario: ' + response.statusText);
33     }
34
35     const data = await response.json();
36     return data; // Devuelve los datos del usuario en JSON
37   }
38   catch (error) {
39     throw new Error('Error de red al obtener los datos del usuario: ' + error.message);
40   }
41 }
42
43
44 export function getTotosUsuarios() {
45   return new Promise((resolve, reject) => {
46     fetch(`${url}/tots-usuaris`)
47       .then(response => {
48         if (response.ok) {
49           return response.json();
50         } else {
51           reject('Error al obtener los usuarios: ' + response.statusText);
52         }
53       })
54       .then(data => {
55         resolve(data);
56       })
57       .catch(error => {
58         reject('Error de red al obtener los usuarios: ' + error.message);
59       });
60   });
61 }

```

Gràcies al communication Manager estan totes les funcions es més fàcil trobar treballar, a l'hora de desplegar també es més ràpid i senzill ja que es comentar una línia i descomentar una altra.

Una altra cosa dins del communication Manager es el fetch a la api de chat gpt i el props que vam crear per que no fos un chat i fos un entrenador personal, un assessor personal de gimnàs i un dietista.

Captura de la api de rutina:

```
export async function enviarMensajeOpenAIRutina(message, ejercicios, datosUsuario) {
  try {
    const payload = {
      model: 'gpt-3.5-turbo',
      messages: [
        {
          role: 'system',
          content: "Ets una persona que només parla en català i tens prohibit parlar d'alguna cosa que no tingui relació amb el fitness ja que ets un expert en fitness només tens permès parlar de fer rutines." +
            " Si et demanen alguna cosa que no sigui una rutina digues el següent: En aquest apartat només puc donar consells de nutrició i generar rutines. " +
            " Pots donar consells i arguments però fes-ho de forma resumida en unes 2 línies a menys que t'indiquin que volen més informació." +
            " Només pots respondre amb format JSON quan i seguint aquesta estructura: " +
            " { 'id_usuari': '', 'dias': [{dia: '1', ejercicios: [{ 'nom_exercici': '', 'series': '', 'repeticions': '', 'id_exercici': '' }, ... ] }, ... ] } " +
            " Segueix aquesta estructura de JSON pero posa per dia un mínim de 5 exercicis i un màxim de 7 exercicis." +
            " Fes un grup muscular per dia i no repetir exercicis en la mateixa rutina. A no ser que et digui el contrari o algo més concret." +
            " Si en el missatge conte dades específiques o informació afegeix tota l'informació i crea la dieta; si no fes una rutina de 5 dies amb 5 músculs per dia i que sigui variada." +
            " Si en el missatge s'especifica quin tipus de rutina volen i els dies, fes-la com ho diuen. Si falta informació fes 5 dies amb 5 músculs per dia variats" +
            " Afegeix les dades del usuari per fer rutines més personalitzades i també afegeix el id del usuari per posarlo a id_usuari."
        },
        {
          role: 'system',
          content: JSON.stringify(ejercicios)
        },
        {
          role: 'system',
          content: JSON.stringify(datosUsuario)
        },
        {
          role: 'user',
          content: message
        }
      ]
    };

    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Authorization': 'Bearer ${apiKey}',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(payload),
    });

    if (!response.ok) {
      throw new Error('HTTP error! status: ${response.status}');
    }

    const data = await response.json();
    const generatedText = data.choices[0].message.content;

    return generatedText;
  } catch (error) {
    throw new Error('Error al enviar el mensaje a OpenAI: ' + error.message);
  }
}
```

## Api de dieta:

```
export async function enviarMensajeOpenAIDieta(message, alimentos, datosUsuario) {
  try {
    const payload = {
      model: 'gpt-4-turbo',
      messages: [
        {
          role: 'system',
          content: "Eres una persona que solo habla catalán y tienes prohibido hablar de nada que no esté relacionado con el fitness y la nutrición porque eres un experto en nutrición y fitness pero tienes muy prohibido hacer rutinas y dietas. " +
            " Si te piden una rutina di lo siguiente: En esta sección solo puedo dar consejos de nutrición y deporte si quieres generar rutinas ve a la sección de Rutinas y si quieres una dieta en la sección de Dietas. " +
            " Si pides dar consejos y argumentos pero hacerlo de forma resumida en 2 líneas, que no te digan que quieren más información." +
            " Coge los alimentos y toda la información que encuentres en el json de alimentos para crear la dieta." +
            " Sólo puedes responder con formato JSON y siguiendo esta estructura: " +
            " { 'id_usuario': '', 'apats': {apart: { 'platos': { 'nom_plat': '', 'ingredientes': { 'nom_ingredient': '', 'quantitat': '', 'unitat': '' }, 'proteinas': '', 'carbohidrats': '', 'grasas': '', 'calorias': '' }, 'apart': { 'platos': { 'nom_plat': '', 'ingredientes': { 'nom_ingredient': '', 'quantitat': '', 'unitat': '' }, 'proteinas': '', 'carbohidrats': '', 'grasas': '', 'calorias': '' }, 'apart': { 'platos': { 'nom_plat': '', 'ingredientes': { 'nom_ingredient': '', 'quantitat': '', 'unitat': '' }, 'proteinas': '', 'carbohidrats': '', 'grasas': '', 'calorias': '' } } } } } " +
            " Si te piden definición o adelgazar o definir, toma platos que sean para ello; si quieren volumen, ganar mucho músculo o ganar peso toma los platos necesarios para ello. Si no te especifican una dieta neutra." +
            " Haz que las 5 comidas diarias sean: Desayuno, Segundo desayuno, Comida, Merienda y Cena. " +
            " Para cada comida diarla ponle 3 platos para cada una de las 5 comidas. Si te piden algo más concreto hazlo de forma más concreta." +
            " No pongas los platos que tengan el alimento que les da alergia basándose en la información del usuario."
        },
        {
          role: 'system',
          content: JSON.stringify(alimentos)
        },
        {
          role: 'system',
          content: JSON.stringify(datosUsuario)
        },
        {
          role: 'user',
          content: message
        }
      ]
    };

    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Authorization': 'Bearer ${apiKey}',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(payload),
    });

    if (!response.ok) {
      throw new Error('HTTP error! status: ${response.status}');
    }

    const data = await response.json();
    const generatedText = data.choices[0].message.content;

    return generatedText;
  } catch (error) {
    throw new Error('Error al enviar el mensaje a OpenAI: ' + error.message);
  }
}
```

## Api de assessorament:

```
export async function enviarMensajeOpenAI(message) {
  try {
    const payload = {
      model: 'gpt-3.5-turbo',
      messages: [
        {
          role: 'system',
          content: "Eres un experto en fitness y nutrición. Por lo tanto solo puedes hablar de temas relacionados con la salud y el fitness"+
            " Solo puedes hablar en catalan."+
            " Si te piden una dieta o una rutina di lo siguiente: En aquest apartat només puc donar consells de nutrició i assessorarte esportiu."+
            " Los consejos y explicaciones que des hazlos de unas 2 líneas mas o menos, a menos que te indiquen que quieren mas informacion o saber mas.",
        },
        {
          role: 'user',
          content: message
        }
      ]
    };

    const response = await fetch(apiUrl, {
      method: 'POST',
      headers: {
        'Authorization': 'Bearer ${apiKey}',
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(payload),
    });

    if (!response.ok) {
      throw new Error(`HTTP error! status: ${response.status}`);
    }

    const data = await response.json();
    const generatedText = data.choices[0].message.content;

    return generatedText;
  } catch (error) {
    throw new Error(`Error al enviar el mensaje a OpenAI: ` + error.message);
  }
}
```

Vam tenir que aprendre com funcionava la api de chat, també com funcionen el props que ens van donar molts malts de cap ja que afegeixes una cosa i es canviava per complet, treies una coma i et feia una altra rutina o dieta totalment diferent. I ara que es funcional es un còdi que ens sentim orgullosos, hem après algo nou i es funcional.

## ○ Registre

El nostre registre, també es una peça important ja que no es un registre típic avorrit i llarg. Vam buscar la manera de fer un registre agil, bonic, atractiu, en el que la gent no li dongues avorrimet omplir-lo. Vam fer-li cas al Ermengol que va ser qui ens va proposar això, hi ha quedat molt bé. I amb el css atractiu i les funcions de que et puguis saltar moltes de les preguntes per omplir-les més endavant ha quedat molt bé.

Imatges del codi de registre:

```

<template>
  <div class="register-container">

    <div v-if="showStartButton">

      <div class="logo-container">
        
      </div>
      <div class="button-container">
        <button class="button button--secondary" @click="startRegistration">Començar Registre</button>
        <button class="button button--secondary" @click="goToIniciarSesion">Iniciar sessió</button>
      </div>
    </div>

    <div v-else>
      <div class="progress-bar" :style="{ width: progressPercentage + '%' }"></div>

      <div v-if="currentQuestionIndex === registrationQuestions.length" class="loading">
        
      </div>
      <div v-if="currentQuestionIndex < registrationQuestions.length" class="question-container fade-in">
        <div v-if="currentQuestionIndex <= registrationQuestions.length" class="question-counter">
          {{ currentQuestionIndex + 1 }} / {{ totalQuestions }}
        </div>
        <h2 class="registrationQuestion fade-in">{{ registrationQuestions[currentQuestionIndex].question }}</h2>

        <input v-if="registrationQuestions[currentQuestionIndex].inputType === 'textarea'"
          v-model="currentAnswer" placeholder="Escribe tu respuesta"></input>
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'email'"
          v-model="currentAnswer" type="email" placeholder="Correu electrònic" @input="validateEmailInput">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'contrasenya'"
          v-model="currentAnswer" type="password" placeholder="Contrasenya" @input="validatePassword">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'nom_usuari'"
          v-model="currentAnswer" type="text" placeholder="Nom d'usuari" maxlength="20">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'nom'"
          v-model="currentAnswer" type="text" placeholder="Nom" @input="validateNameInput" maxlength="25">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'cognoms'"
          v-model="currentAnswer" type="text" placeholder="Cognoms" @input="validateNameInput" maxlength="80">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'altura'"
          v-model="currentAnswer" type="text" placeholder="Altura (cm)" @input="validateAlturaInput"
          maxlength="4">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'pes'"
          v-model="currentAnswer" type="text" placeholder="Pes (kg)" @input="validatePesInput" maxlength="6">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'telefon'"
          v-model="currentAnswer" type="tel" placeholder="Numero de telèfon" @input="validateTelefonInput">
        <input v-else-if="registrationQuestions[currentQuestionIndex].inputType === 'data_naixement'"
          v-model="currentAnswer" type="date" @input="validateDate">
        <div v-if="registrationQuestions[currentQuestionIndex].inputType === 'genre'">
          <div class="gender-options">
            <div v-for="(option, index) in registrationQuestions[currentQuestionIndex].respuesta"
              :key="index" class="gender-option">
              <input type="radio" :id="'option' + index" :value="option" v-model="currentAnswer">
              <label :for="'option' + index" @click="selectGenderOption(option)">{{ option }}</label>
            </div>
          </div>
        </div>

        <div v-if="showErrorMessage" class="error-message">
          {{ errorMessage }}
        </div>
        <div v-if="registrationQuestions[currentQuestionIndex].inputType === 'contrasenya'"
          class="error-message">{{ passwordErrorMessage }}</div>
        <div v-if="registrationQuestions[currentQuestionIndex].inputType === 'data_naixement'"
          class="error-message">{{ dataErrorMessage }}</div>
      </div>
    </div>
  </div>

```

```
<script>
import { useUsuariPerFilStore } from '@stores/index'

export default {
  data() {
    return {
      showStartButton: true,
      checkingEmail: false,
      errorMessage: "",
      showErrorMessage: false,
      currentQuestionIndex: 0,
      currentAnswer: "",
      selectedOptions: [],
      phoneNumberValid: false,
      passwordErrorMessage: "",
      dateErrorMessage: "",
      userData: {
        email: "",
        contrasenya: "",
        nom_usuari: "",
        nom: "",
        cognoms: "",
        data_naixement: "",
        genere: "",
        pes: "",
        altura: "",
        telefon: "",
        registre: false,
      },
      totalQuestions: 0,
      progressPercentage: 0,

      registrationQuestions: [
        {
          question: "Quin es el teu correu electrònic?",
          inputType: 'email',
          required: true,
        },
        {
          question: "Quina es la teva contrasenya?",
          inputType: 'contrasenya',
          required: true,
        },
        {
          question: "Quin es el teu nom d'usuari?",
          inputType: 'nom_usuari',
          required: true,
        },
        {
          question: "Quin es el teu nom?",
          inputType: 'nom',
          required: true,
        },
        {
          question: "Quins son els teus cognoms?",
          inputType: 'cognoms',
          required: true,
        },
      ],
    }
  },
}
```

```

computed: {
  progressPercentage() {
  },
},
methods: {
  startRegistration() {
    this.showStartButton = false;
    this.totalQuestions = this.registrationQuestions.length; // Establecer el número total de preguntas
  },
  goToIniciarSesion() {
    this.$router.push('/');
  },
  async seguentPregunta() {
    // Obtener la pregunta actual
    const currentQuestion = this.registrationQuestions[this.currentQuestionIndex];

    // Verificar si la pregunta actual es requerida y si la respuesta está vacía
    if (currentQuestion.required && this.currentAnswer === "") {
      // Mostrar un mensaje de error indicando que la pregunta es requerida
      this.showErrorMessage = true;
      this.errorMessage = "Aquesta pregunta és requerida. Si us plau, respon abans de continuar.";
      return;
    }

    // Verificar si la respuesta actual está vacía
    if (this.currentAnswer === "") {
      // Mostrar un mensaje de error indicando que la respuesta es requerida
      this.showErrorMessage = true;
      this.errorMessage = "Aquest camp no pot estar buit. Si us plau, respon abans de continuar.";
      return;
    }

    // Validar la contraseña si es la pregunta de contraseña
    if (currentQuestion.inputType === 'contrasenya') {
      const isPasswordValid = this.validatePassword();
      if (!isPasswordValid) {
        // Mostrar mensaje de error si la contraseña no es válida
        return;
      }
    }

    // Validar la fecha si es la pregunta de fecha de nacimiento
    if (currentQuestion.inputType === 'data_naixement') {
      const isDateValid = this.validateDate();
      if (!isDateValid) {
        // Mostrar mensaje de error si la fecha no es válida
        return;
      }
    }

    // Validar el número de teléfono si es la pregunta de teléfono
    if (currentQuestion.inputType === 'telefon') {
      this.validateTelefonInput(); // Validar el número de teléfono
      if (this.showErrorMessage) {
        // Mostrar mensaje de error si el número de teléfono no es válido
        return;
      }
    }
  }
}

```

```
// Validar el correo electrónico si es la pregunta de correo electrónico
if (currentQuestion.inputType === 'nom_usuari') {

    this.checkingEmail = true;
    // Realizar la comprobación de correo electrónico solo si la respuesta no está vacía
    const response = await fetch('http://fithub.daw.inspedralbes.cat/back/public/api/comprovarnomusuari', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
        },
        body: JSON.stringify({
            nom_usuari: this.currentAnswer
        }),
    });

    this.checkingEmail = false;

    // Convertir la respuesta a formato JSON
    const responseData = await response.json();

    // Si el correo ya existe, mostrar el mensaje de error y no avanzar
    if (responseData.status === 1) {
        this.showErrorMessage = true;
        this.errorMessage = responseData.message;
        return;
    }
}

if (currentQuestion.inputType === 'email') {
    this.validateEmailInput(); // Validar el correo electrónico
    if (this.showErrorMessage) {
        // Mostrar mensaje de error si el correo electrónico no es válido
        return;
    } else {
        // Realizar la comprobación de correo electrónico solo si la respuesta no está vacía
        this.checkingEmail = true;
        const response = await fetch('http://fithub.daw.inspedralbes.cat/back/public/api/comprovaremail', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json',
            },
            body: JSON.stringify({
                email: this.currentAnswer
            }),
        });

        this.checkingEmail = false;

        // Convertir la respuesta a formato JSON
        const responseData = await response.json();

        // Si el correo ya existe, mostrar el mensaje de error y no avanzar
        if (responseData.status === 1) {
            this.showErrorMessage = true;
            this.errorMessage = responseData.message;
            return;
        }
    }
}
```

DEBUG CONSOLE TERMINAL PORTS

```

// Si la pregunta no es requerida, o si es requerida pero tiene respuesta, proceder a la siguiente pregunta
// Guardar la respuesta actual en el objeto de datos del usuario
this.userData[currentQuestion.inputType] = this.currentAnswer;

// Incrementar el índice de la pregunta
this.currentQuestionIndex++;

// Reiniciar la respuesta actual
this.currentAnswer = "";
this.showErrorMessage = false;

// Verificar si todas las preguntas han sido respondidas
if (this.currentQuestionIndex === this.registrationQuestions.length) {
    // Si todas las preguntas han sido respondidas, establecer registro en true
    this.userData.registre = true;

    // Realizar el registro del usuario
    await this.registerUser();
}
},

async saltarPregunta() {
    // Obtener la pregunta actual
    const currentQuestion = this.registrationQuestions[this.currentQuestionIndex];
    this.currentAnswer = "";

    // Verificar si la pregunta actual es requerida y si la respuesta está vacía
    if (currentQuestion.required) {
        // Mostrar un mensaje de error indicando que la pregunta es requerida
        this.showErrorMessage = true;
        this.errorMessage = "Aquesta pregunta és requerida. Si us plau, respon abans de continuar";
        return;
    }

    // Si hay una respuesta, guardarla en el objeto de datos del usuario
    if (this.currentAnswer.trim() !== "") {
        this.userData[currentQuestion.inputType] = this.currentAnswer;
    }

    // Incrementar el índice de la pregunta
    this.currentQuestionIndex++;

    // Reiniciar la respuesta actual
    this.currentAnswer = "";
    this.showErrorMessage = false;

    // Verificar si todas las preguntas han sido respondidas
    if (this.currentQuestionIndex === this.registrationQuestions.length) {
        // Si todas las preguntas han sido respondidas, establecer registro en true
        this.userData.registre = false;

        // Realizar el registro del usuario
        await this.registerUser();
    }
},

```

## ○ Chat Rutina o Dieta:

El codi del nostres chat Rutina o Dieta ens sentim contents ja que hem aconseguit fer com un voiceflow que tracta que hi han uns botons i depèn del que seleccions et surten més batons o no. Segons la seqüència que has seleccionat envia un missatge amb tot lo que has seleccionat i el chat et fa lo que li demanes.

Aquí esta el codi:



```
<!-- Opciones de respuesta como botones -->
<div v-if="currentOptions && showOptions" class="botones-preseleccionados">
  <button v-for="(option, key) in currentOptions" :key="key" @click="handleOptionSelect(key)">
    {{ key }}
  </button>
</div>
```

```
const arbrePreguntes = {
  pregunta: "Quin tipus de rutina vols?",
  opciones: {
    Hipertrofia: {
      pregunta: "Quants dies prefereixes entrenar a la setmana?",
      opciones: {
        "3": "Rutina d'hipertrofia de 3 dies.",
        "4": "Rutina d'hipertrofia de 4 dies.",
        "5": "Rutina d'hipertrofia de 5 dies.",
        "6": "Rutina d'hipertrofia de 6 dies.",
      }
    },
    Calistenia: {
      pregunta: "Quants dies prefereixes entrenar a la setmana?",
      opciones: {
        "3": "Rutina de calistenia de 3 dies.",
        "4": "Rutina de calistenia de 4 dies.",
        "5": "Rutina de calistenia de 5 dies.",
        "6": "Rutina de calistenia de 6 dies.",
      }
    },
    Equilibrada: {
      pregunta: "Quants dies prefereixes entrenar a la setmana?",
      opciones: {
        "3": "Rutina equilibrada de 3 dies.",
        "4": "Rutina equilibrada de 4 dies.",
        "5": "Rutina equilibrada de 5 dies. ",
        "6": "Rutina equilibrada de 6 dies.",
      }
    }
  }
};
```

arbre de preguntes i respostes dieta:

```
const arbrePreguntes = {
  pregunta: "Quin tipus de dieta vols?",
  opcions: {
    Volum: {
      pregunta: "Quants àpats prefereixes fer al dia per a una dieta de volum?",
      opcions: {
        "4": "Dieta de volum de 4 àpats esmorzar, dinar, berenar, sopar",
        "5": {
          pregunta: "Vols incloure un segon esmorzar o un post-entrenament?",
          opcions: {
            "Segon esmorzar": "Dieta de volum de 5 apats Esmorzar, Segon esmorzar, Dinar, Berenar, Sopar",
            "Post-entrenament": "Dieta de volum de 5 apats Esmorzar, Dinar, Berenar, Post-entrenament, Sopar"
          }
        },
        "6": "Dieta de volum de 6 apats Esmorzar, Segon esmorzar , Dinar, Berenar, Post-entrenament, Sopar"
      }
    },
    Definició: {
      pregunta: "Quants àpats prefereixes fer al dia per a una dieta de definició?",
      opcions: {
        "4": "Dieta de definició de 4 apats Esmorzar, Dinar, Berenar, Sopar",
        "5": {
          pregunta: "Vols incloure un Segon esmorzar o un Post-entrenament?",
          opcions: {
            "Segon esmorzar": "Dieta de definició amb 5 apats Esmorzar, Segon Esmorzar, Dinar, Berenar, Sopar",
            "Post-entrenament": "Dieta de definició amb 5 apats: Esmorzar, Dinar, Post-entrenament, Berenar, Sopar."
          }
        },
        "6": "Dieta de definició amb 6 apats: Esmorzar, Segon Esmorzar, Dinar, Post-entrenament, Berenar, Sopar"
      }
    },
    Equilibrada: {
      pregunta: "Quants àpats prefereixes al dia per a una dieta equilibrada?",
      opcions: {
        "4": "Dieta equilibrada amb 4 àpats",
        "5": {
          pregunta: "Vols incloure un segon esmorzar o un post-entrenament?",
          opcions: {
            "Segon esmorzar": "Dieta equilibrada amb 5 apats: Esmorzar, Segon Esmorzar, Dinar, Berenar, Sopar",
            "Post-entrenament": "Dieta equilibrada amb 5 apats: Esmorzar, Dinar, Post-entrenament, Berenar, Sopar"
          }
        },
        "6": "Dieta equilibrada amb 6 àpats"
      }
    }
  }
}
};
```

```
handleOptionSelect(optionKey) {
  let nextStep = this.currentOptions[optionKey];

  // Comprobar si hay un siguiente nivel de opciones
  if (nextStep && typeof nextStep === 'object' && nextStep.pregunta) {
    // Actualizar la pregunta y opciones actuales si hay más preguntas
    this.currentQuestion = nextStep.pregunta;
    this.currentOptions = nextStep.opciones || {};
    // Añadir la respuesta del asistente al chat
    this.chatMessages.push({
      role: 'assistant',
      content: this.currentQuestion
    });
  } else {
    // Manejar el final del árbol de preguntas
    this.currentQuestion = nextStep; // Esta sería la respuesta final
    this.currentOptions = {};
    // Establecer el mensaje final en `message` para ser enviado
    this.message = nextStep;
    // Añadir al chat y preparar para enviar
    this.chatMessages.push({
      role: 'assistant',
      content: this.currentQuestion
    });
    // Opcional: Llamar a enviarMensaje directamente si se desea enviar inmediatamente
    this.enviarMensaje();
  }
},
```

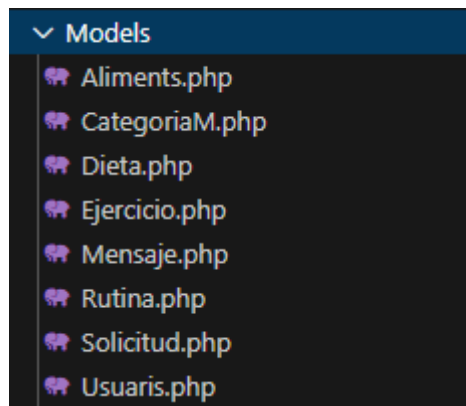
Definim una estructura amb arbre amb diferents opcions, aquestes opcions es mostren en format botó. Si hi ha més opcions mostra la següent pregunta amb les seves opcions i si es l'última envia el la resposta final com a missatge.

## 6. Documentació de codi backend

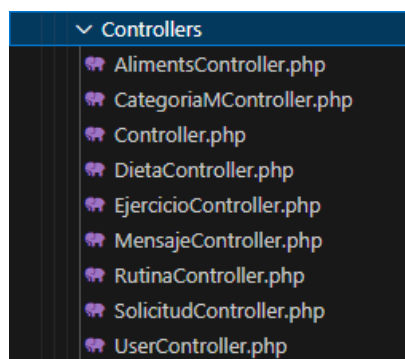
El nostre projecte hem utilitzat tres tecnologies que son les següents: Nuxt, Laravel i també hem utilitzat la API de OpenAI.

### ○ **Laravel:**

- **Descripció:** En Laravel vam utilitzar com a back, hem fet amb aquesta tecnologia una Api per fer diferents funcions. La utilitzem per fer gestió d'usuaris, gestió de rutina, també per fer gestió de dietes, gestió de sol·licituds o gestió de missatges.
- **Estructura del codi:**
  - **app/:** Conté el codi font del projecte, inclòs models, controlador i també email.
  - **Models:** En el models trobarem la informació sobre les dades de la base de dades, trobarem important per fer el insertar.



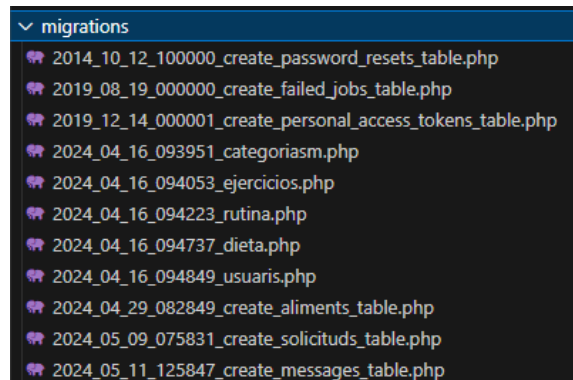
- **Http:** En la carpeta controller es on trobem tota la informació i funcions de la api de cadascun del controladors, normalment les funcions serveixen per crear, eliminar, modificar i mostrar.



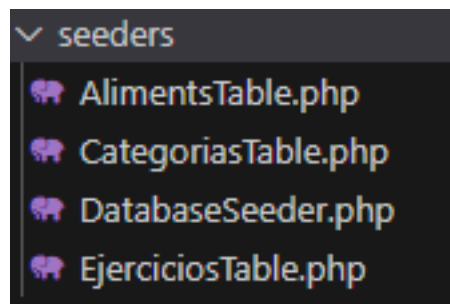
- **Mail:** En la carpeta trobarem el constructor on enviem el missatge també enviem la informació d'usuari per mostrar informació de l'usuari en el correu.

- **database/:** Conté les migracions i els seeds per la base de dades.

- **Migrations:** Aquí trobem les taules que es troben en la base de dades, amb tots el camps de la taula.

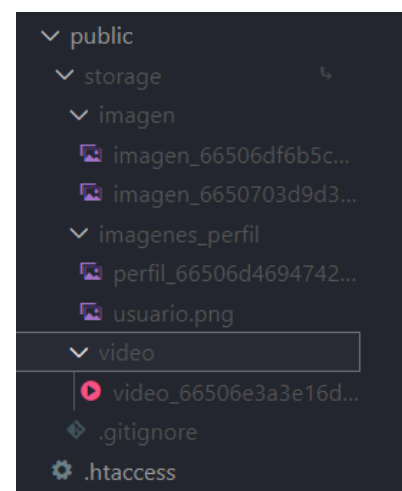


- **Seeders:** Aquí trobem la informació on insertarem tota la informació i insertar en la taules de la base de dades.



- **public/storage/:** Trobem aquí carpetes que es on trobem les imatges, vídeos o imatge de perfil, que son les carpetes on es guarda.

- **imagen:** Trobem totes les imatges que es guarden de les imatges que trobem en el chat.
- **imagenes\_perfil:** Trobem les imatges que es posa l'usuari a l'hora d'insertar la seva foto de perfil.
- **video:** Trobem tots el vídeos que es guarden del chat, quan comparteixen videos.



- **resources/views/email/**: Aquí el que trobem es la vista que es mostrar en els emails un cop registrat.
- **routes/**: En aquesta carpeta trobarem diferents arxius i en l'api en el fitxer, trobem la funcions de l'aplicació.
  - **UserController**: Aquí trobareu les sortides de la api amb el nom de les funcions i el controlador que és d'usuaris.

```
Route::get('/tots-usuaris', [UserController::class, 'listarUsuaris']);
Route::delete('/eliminar-usuari/{id}', [UserController::class, 'eliminarUsuario']);
Route::get('/usuaris', [UserController::class, 'getUsers']);
Route::get('/usuari/{id}', [UserController::class, 'mostrarUsuario']);
Route::put('/editar-usuari/{id}', [UserController::class, 'editarUsuario']);
Route::get('/chatUsuaris/{id}', [UserController::class, 'getAmics']);
Route::get('/usuarios-excepto/{idUser}', [UserController::class, 'mostrarUsuariosExceptoYo']);
Route::post('/registre', [UserController::class, 'registre']);
Route::post('/loguejat', [UserController::class, 'loguejat']);
Route::post('/deslojgat', [UserController::class, 'deslojgat']);
Route::post('/comprovaremail', [UserController::class, 'comprovarCorreoUsuario']);
Route::post('/comprovarnomusuari', [UserController::class, 'comprovarNomUsuario']);
```

- **SolicitudController**: Aquí trobareu les sortides de la api amb el nom de les funcions i el controlador que es de solicitud.

```
Route::post('/enviar-solicitud', [SolicitudController::class, 'enviarSolicitud']);
Route::get('/mostrar-solicitudes/{id}', [SolicitudController::class, 'MostrarUsuarioSolicitudes']);
Route::get('/mostrar-solicitudes-enviadas/{id}', [SolicitudController::class, 'MostrarUsuarioSolicitud']);
Route::post('/aceptar-solicitud/{id}', [SolicitudController::class, 'AceptarAmigo']);
Route::delete('/eliminar-solicitud/{id}', [SolicitudController::class, 'RechazarAmigo']);
```

- **MensajeController**: Aquí trobareu les sortides de la api amb el nom de les funcions i el controlador que es de missatge.

```
Route::post('/enviar-mensaje/{usuari1}/{usuari2}', [MensajeController::class, 'enviarMensaje']);
Route::get('/missatges/{userId1}/{userId2}', [MensajeController::class, 'mostrarMensajeEntreEllos']);
Route::get('/ultim-missatge/{userId1}/{userId2}', [MensajeController::class, 'mostrarUltimoMensajeEntreEllos']);
```

- **EjercicioController**: Aquí trobareu la sortida de la api amb el nom de la funció i el controlador que es d'exercici.

```
Route::get('/ejercicios', [EjercicioController::class, 'getEjercicios']);
```

- **RutinaController**: Aquí trobareu les sortides de la api amb el nom de les funcions i el controlador que es rutina.

```
Route::get('/rutina/{id}', [RutinaController::class, 'getRutina']);
Route::post('/guardar-rutina', [RutinaController::class, 'store']);
Route::delete('/eliminar-rutina/{id}', [RutinaController::class, 'destroyTodo']);
Route::delete('/eliminar-rutina-hoy/{id}', [RutinaController::class, 'destroyToday']);
Route::delete('/rutinas/{id_usuario}/{fecha}', [RutinaController::class, 'destroyByDate']);
Route::post('/descargarRutina', [RutinaController::class, 'saveRoutines']);
```

- **AlimentsController**: Aquí trobareu la sortida de la api amb el nom de la funció i el controlador que es d'aliments.

```
Route::get('/aliments', [AlimentsController::class, 'index']);
```

- **DietaController**: Aquí trobareu les sortides de la api amb el nom de les funcions i el controlador que es dieta.

```
Route::get('/dieta/{id}', [DietaController::class, 'getDietas']);
Route::post('/guardar-dieta', [DietaController::class, 'store']);
Route::delete('/eliminar-dieta-hoy/{id}', [DietaController::class, 'destroyToday']);
Route::delete('/dietas/{id_usuario}/{fecha}', [DietaController::class, 'destroyByDate']);
```

## 7. Disseny

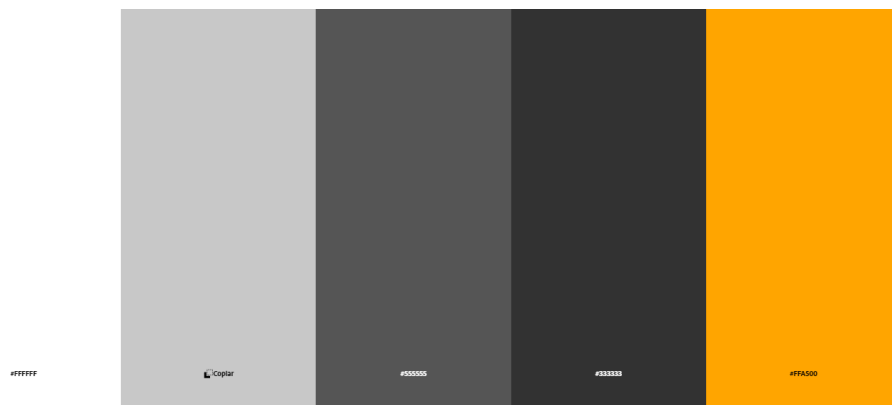
### Imatge corporativa

Pel que fa al disseny de la nostra aplicació, hem intentat seguir una estètica innovadora, ja que la innovació és un dels apartats claus del nostre projecte. Això es veu reflexat en l'estètica minimalista, amb colors clars predominants al fons de la pàgina, y elements destacats amb els colors de la nostra marca.



El nostre logo, amb les seves variacions, consta de dos elements principals, el text amb la paraula FitHub i la dumbell característica que també hem fet servir en les animacions de càrrega del registre o la barra de navegació.

Aquesta és la paleta de colors que hem utilitzat:



Com a directives generals per al css, hem fet en la mesura del possible que totes les pàgines siguin el més responisve possible, per a que es vegin correctament en qualsevol pantalla i dispositiu, centrant-nos més en la part mòvil, ja que la nostra aplicació no estava pensada per desktop.

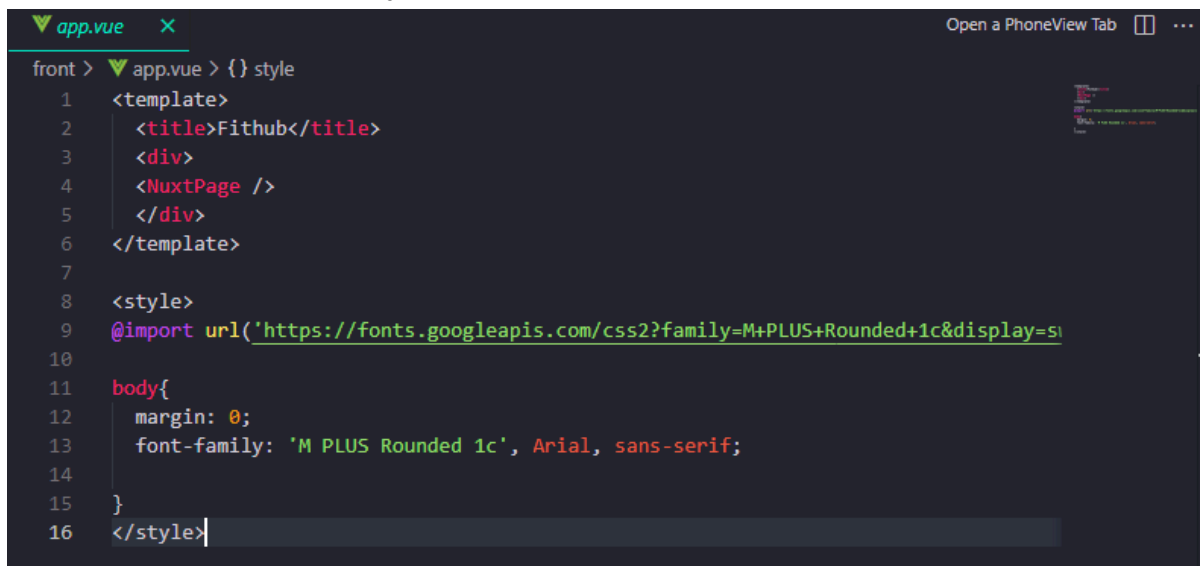


## Tipografia

En el css, hem utilitzat un estil general per al body, on hem tret el margin del body y hem carregat la nostra tipografia, que és la següent:

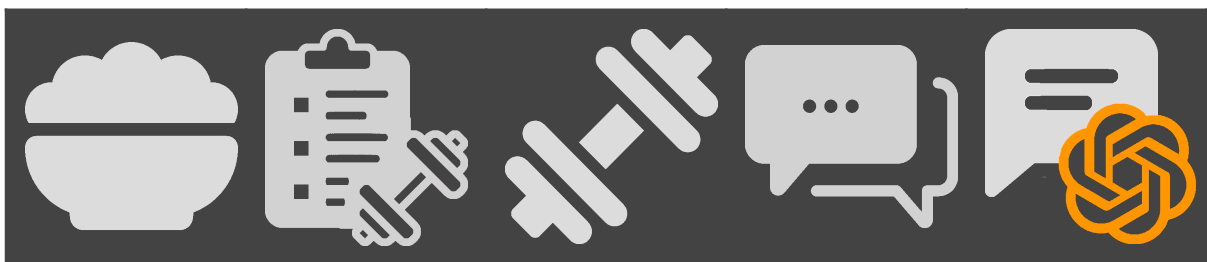


És una tipografia de Google Fonts de lliure ús, que hem pensat que era la que més s'adaptava a l'estètica del projecte.

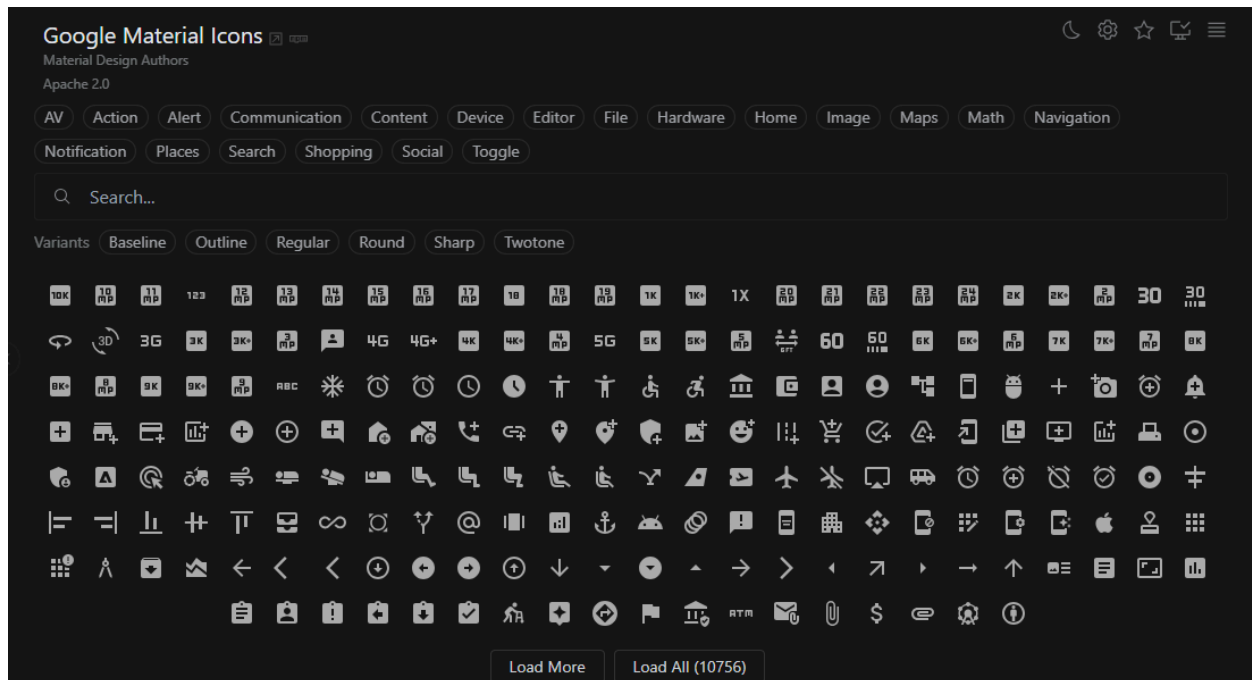


## Icones

Respecte a les icones, els hem obtingut de dues fonts, d'una banda, les icones de la barra de navegació són de disseny propi:

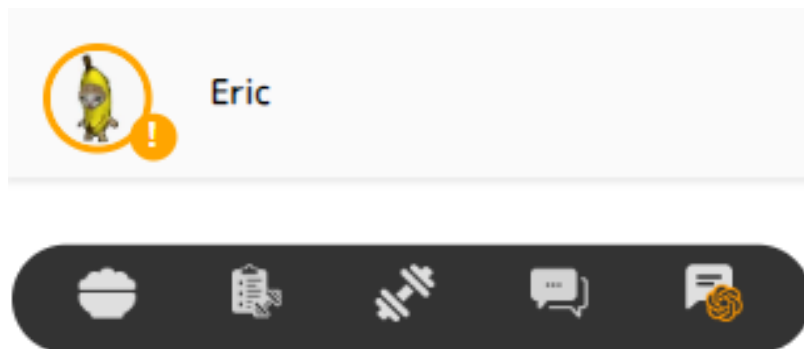


La resta d'ícones que apareixen, les hem obtingut del pack Google Material Icons, de la llibreria d'ícones <https://icons.js.org/>



## Pàgines

El layout de pàgines que hem triat, consta sempre dels mateixos dos elements en totes les pàgines, que són la barra de navegació per poder navegar per totes les pàgines de l'aplicació i un header amb la foto de perfil i el nom de l'usuari.:



### *CSS del header:*

```
<style scoped>
a{
  text-decoration: none;
  color: black;
}

.imgContainer {
  display: grid;
  grid-template-columns: .6fr 1fr;
  margin: auto;
}

.user-info {
  position: relative;
  margin-left: 20px;
  display: grid;
  grid-template-columns: 1.5fr .2fr;
  padding-bottom: 20px;
}

.user-icon-container {
  margin-top: 20px;
  width: 50px; /* Ajustar el tamaño según sea necesario */
  height: 50px; /* Ajustar el tamaño según sea necesario */
  border-radius: 50%; /* Hacer el contenedor redondo */
  overflow: hidden; /* Ocultar cualquier parte de la imagen que se
desborde del contenedor */
}

.incomplete-profile{
  margin-top: 20px;
  width: 50px; /* Ajustar el tamaño según sea necesario */
  height: 50px; /* Ajustar el tamaño según sea necesario */
  border-radius: 50%; /* Hacer el contenedor redondo */
  overflow: hidden; /* Ocultar cualquier parte de la imagen que se
desborde del contenedor */
  border: 4px solid #FFA500;
}

.user-icon {
```

```

        width: 100%; /* Ajustar la imagen al 100% del ancho del contenedor
    */
        height: 100%; /* Ajustar la imagen al 100% de la altura del
    contenedor */
        object-fit: cover; /* Escalar la imagen para que llene el
    contenedor sin distorsionarse */
    }

.alert-sign {
    position: absolute;
    bottom: 10%;
    left: 15%;
    transform: translateX(5%);
    width: 25px;
    height: 25px;
    background-color: #FFA500;
    border-radius: 50%;
    display: flex;
    justify-content: center;
    align-items: center;
    font-family: Arial, sans-serif;
    color: white;
    font-size: 1.2rem;
    font-weight: bolder;
}

h1 {
    padding: 20px;
    font-size: 1.1rem;
}

.header-container {
    background-color: #fafafa;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    margin: auto;
    margin-top: 0;
    position: relative;
    margin-bottom: 20px;
    width: 100%;
    display: grid;
    grid-template-columns: .1fr 1fr 2fr;
    align-items: center;
}

```

```
.amics{
    text-align: right;
    margin-right: 20px;
}

.menu-button {
    width: 30px; /* Ajustar el ancho según sea necesario */
    height: 30px; /* Ajustar la altura según sea necesario */
    border: none;
    background-color: transparent; /* Botón transparente */
    cursor: pointer;
    transition: transform 0.3s ease; /* Transición para animar el botón */
}

.menu-button img{
    width: 30px;
    height: 30px;
}

.menu-desplegable {
    position: absolute;
    top: 40px; /* Ajustar la posición según sea necesario */
    right: 10px; /* Ajustar la posición según sea necesario */
    background-color: white;
    border: 1px solid #ccc;
    border-radius: 4px;
    padding: 5px;
    z-index: 999; /* Asegura que esté por encima de otros elementos */
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1); /* Sombra */
    transition: max-height 0.3s ease; /* Transición para animar el
despliegue del menú */
    overflow: hidden;
}

.slide-enter-active, .slide-leave-active {
    transition: max-height 0.3s ease; /* Añade una transición suave */
}

.slide-enter, .slide-leave-to {
    max-height: 200px; /* Altura máxima del menú */
}
```

```
.menu-desplegable button {
  display: block;
  width: 100%;
  text-align: left;
  padding: 8px 10px;
  border: none;
  background-color: transparent;
}

.menu-desplegable button:hover {
  background-color: #f0f0f0;
}
</style>
```

### *CSS de la barra de navegació:*

```
<style scoped>
.navbar {
  background-color: #333;
  color: white;
  width: 90%;
  padding: 10px; /* Modificado para ajustar el espacio vertical */
  margin: auto;
  margin-bottom: 20px;
  display: flex;
  justify-content: center; /* Centra los elementos horizontalmente */
  position: sticky;
  top: 0;
  border-radius: 30px;
  z-index: 1000;
}

.navbar-container {
  display: flex;
```

```
justify-content: space-between;
align-items: center;
}

.navbar-brand {
  font-size: 1.5rem;
  margin-left: 25px;
}

.navbar-links {
  display: flex;
  justify-content: center; /* Centra los enlaces horizontalmente */
  margin: auto;
}


.navbar-link {
  color: white;
  text-decoration: none;
  margin: 0 15px; /* Añade espacio entre los enlaces */
  text-align: center;
}

.navbar-link:hover {
  text-decoration: underline;
}

img{
  width: 60%;
  text-align: center;
}
</style>
```

*Les pantalles principals de la nostra aplicació són:*

- **El login / registre:**

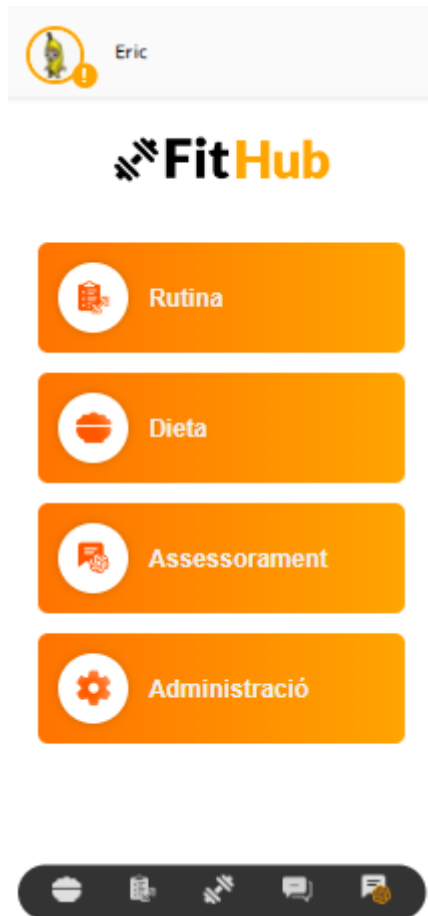


The image shows a mobile application screen for login and registration. The background is a solid orange color. At the top center, the FitHub logo is displayed in white. Below the logo, there is a white rectangular box containing the login and registration forms. Inside this box, the text 'Correu electrònic:' is followed by a text input field with the placeholder 'Correu electrònic'. Below that, the text 'Contrasenya:' is followed by a text input field with the placeholder 'Contrasenya'. At the bottom of the white box, there are two orange buttons: 'Iniciar sessió' and 'Registre'.

Aquesta és la primera pantalla que veuen els usuaris, i consta del següent codi, amb un contenidor principal, amb dos contenidors secundaris, un per al logo i un altre per al formulari:



- La pantalla principal / home:



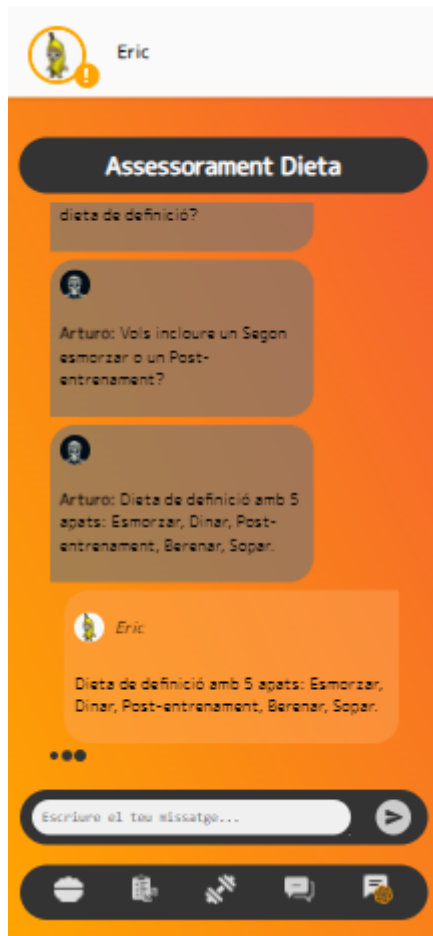
Aquesta és la pantalla principal, en fer login o registrar-se és el primer que es veu. Consta de 3 botons principals, que dirigeixen als 3 apartats claus de la nostra aplicació (Rutina, Dieta, Assessorament). El botó d'administració només és visible per usuaris que tinguin permisos d'administració. Els botons estan continguts dins un contenidor que ocupa el 85% de la pàgina, centrat. A la part superior trobem un contenidor per al logo.

- Pantalla de rutina:



La pantalla de rutina és on es mostren els exercicis de la nostra rutina. Estàn continguts en un contenidor principal que ocupa el 70% de la pàgina. Estan ordenats per dia, en format targeta, on cada targeta és un exercici amb la seva imatge i altres dades.

- Pantalla de chat IA:



Aquesta és una de les pantalles més complexes pel que fa a disseny, ja que hem hagut de diferenciar els missatges de l'usuari i els de la IA, i per tant hem hagut de generar diversos tipus de contenidors per a cada tipus de missatge. L'estructura que segueix és: un contenidor principal que conté tota la pàgina, un contenidor que conté el chat, que ocupa el 90% de l'ample de la pàgina, amb la propietat `overflow-y: auto`, que fa que el chat per molts missatges que tinguem, no es surti del seu contenidor i per tant que la pàgina no ocupi l'alçada de tot el chat. Això és necessari per a que la pàgina no sigui excessivament llarga, i que sempre sigui visible la barra d'introducció de text i la barra de navegació. El contenidor del chat està distribuït en columnes i centrat mitjançant `display: flex` i `flex-direction: column`. Després els missatges es diferencien entre missatge-usuari o missatge-assistent, i contenen un altre contenidor a dins que és el contingut del missatge.

## Exemples de codi css:

### - Pantalla login / registre:

```
<style scoped>
/* Estilos de la página de inicio de sesión */

html,
body {
  margin: 0;
  padding: 0;
  height: 100%;
}

body {
  /* Establecer la fuente predeterminada */
  background-color: #f0f0f0;
  /* Color de fondo */
}

.login-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  height: 100vh;
  background-color: #FFA500;
  /* Orange color */
}

.logo-container {
  margin-bottom: 1.5rem;
  /* Equivalente a 6px */
}

.logo-image {
  width: 300px;
  height: 300px;
}

.form-container {
  background-color: #FFF;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  border-radius: 0.5rem;
```

```
border: 1px solid #000;
padding: 2rem;
margin-bottom: 2.5rem;
/* Equivalente a 10px */
width: 300px;
}

.form-field {
margin-bottom: 1rem;
/* Equivalente a 4px */
}

.form-label {
display: block;
color: #333;
/* Text color */
font-size: 1.5rem;
/* Equivalente a 16px */
font-weight: bold;
margin-bottom: 0.5rem;
/* Equivalente a 2px */
}

.form-input {
width: 100%;
padding: 0.5rem;
border: 1px solid #ccc;
/* Light gray border */
border-radius: 0.25rem;
font-size: 1rem;
/* Equivalente a 14px */
color: #333;
/* Text color */
outline: none;
}

.button-container {
display: flex;
flex-direction: column;
/* Botones uno debajo del otro */
align-items: center;
}
```

```
.button {
  margin-top: 0.5rem;
  /* Espacio entre botones */
  padding: 0.5rem 1rem;
  border: none;
  border-radius: 0.25rem;
  cursor: pointer;
  transition: background-color 0.3s ease;
  width: 100%;
}

.button--primary {
  background-color: #FFA500;
  /* Orange color */
  color: #FFF;
  /* Text color */
  font-weight: bold;
  font-size: 1rem;
  /* Equivalente a 14px */
}

.button--secondary {
  background-color: #FF7F00;
  /* Darker orange color */
  color: #FFF;
  /* Text color */
  font-weight: bold;
  font-size: 1rem;
  /* Equivalente a 14px */
}

.error {
  border-color: red;
}

.form-error {
  color: red;
  font-size: 0.8rem;
  margin-top: 0.5rem;
}
</style>
```

- **Pantalla home:**

```
<style scoped>
/* Estilos de los divs en el componente */
html,
body {
  margin: 0;
  padding: 0;
  height: 100%;
  overflow-x: hidden;
  /* Evita el desplazamiento horizontal */
}

.page-enter-active,
.page-leave-active {
  transition: all 0.4s;
}

.page-enter-from,
.page-leave-to {
  opacity: 0;
  filter: blur(1rem);
}

.flex-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  height: 100vh;
  width: 100%;
  margin: auto;
  /* Mínimo 100% de la altura de la ventana */
  background-color: #FFF;
  position: relative;
}

.logo {
  margin-top: 20px;
  width: 50%;
  display: flex;
  justify-content: center;
}
```

```
.logo img {
  width: 100%;
}

.button-container {
  display: grid;
  grid-template-columns: 1fr;
  grid-gap: 20px;
  width: 85%;
  margin: auto;
  /* Empuja hacia abajo los botones */
}

.large-button {
  background-image: linear-gradient(to right, #ff7300, #FFA500);
  color: #f0f0f0;
  border: none;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 1em;
  font-weight: bold;
  cursor: pointer;
  border-radius: 10px;
  width: 100%;
  margin: auto;
  padding: 20px;
  display: flex;
  align-items: center;
}

.button-content {
  display: grid;
  grid-template-columns: 1fr 3fr;
  align-items: center;
  width: 100%;
}

.img-container {
  width: 70px;
  height: 70px;
  display: flex;
  align-items: center;
  justify-content: center;
}
```



```

background-color: white;
border-radius: 50%;
box-shadow: 0 0 10px 0 rgba(0, 0, 0, 0.2);
margin-right: 20px;
}

.img-container img {
  width: 50%;
}

.large-button h1 {
  font-size: 1.5em;
  margin: 0;
  word-wrap: break-word;
  text-align: left;
}

navBar {
  position: fixed;
  width: 100%;
}
</style>

```

#### - Pantalla rutina:

```

<style scoped>
/* Estilos de los divs en el componente */
html,
body {
  margin: 0;
  padding: 0;
  height: 100vh;
  overflow-x: hidden;
  /* Evita el desplazamiento horizontal */
}

/* Timer CSS */

.time-adjust {
  display: flex;
  align-items: center;
}

```

```
.time-adjust-button {
    width: 20px;
    height: 20px;
    font-size: 14px;
    margin-left: 5px;
    cursor: pointer;
}

.timer-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 20px;
    /* Espaciado superior */
    margin-bottom: 20px;
    /* Espaciado inferior */
}

.timer {
    text-align: center;
}

.time-adjust-container {
    display: flex;
    justify-content: center;
}

.timer-buttons {
    margin-top: 10px;
    /* Espaciado entre el temporizador y los botones */
}

/* ////////////////////////////////////// */

.series-button:active {
    background-color: #555;
    /* Cambiar este valor al color deseado */
    color: #fff;
    /* Cambiar el color del texto para asegurar la legibilidad */
}

.modal {
```

```
position: fixed;
top: 0;
left: 0;
width: 100%;
height: 100%;
background-color: rgba(0, 0, 0, 0.5);
/* Fondo oscuro semitransparente */
display: flex;
justify-content: center;
align-items: center;
}

.modal-content {
  background-color: #fff;
  padding: 20px;
  border-radius: 10px;
  max-width: 80%;
  max-height: 80%;
  overflow: auto;
}

.close {
  position: absolute;
  top: 10px;
  right: 10px;
  cursor: pointer;
}

.arrow {
  width: 50px;
  height: 50px;
  margin: auto;
  color: #000;
}

.flex-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  height: 100vh;
  /* Mínimo 100% de la altura de la ventana */
  background-color: #FFF;
}
```

```
.flex-container h1 {
  margin-top: -5px;
  margin-bottom: 3px;
  font-size: 36px;
  font-weight: bold;
}

.main-content {
  flex-grow: 1;
  overflow-y: auto;
  /* Habilita el scroll si el contenido es más grande que la
  ventana */
  padding-top: 10px;
  /* Altura del header */
  padding-bottom: 50px;
  /* Altura del navBar */
  text-align: center;
}

.exercise-list {
  display: grid;
  grid-template-columns: 1fr;
  gap: 20px;
  padding: 20px;
  margin: auto;
  text-align: center;
  border-radius: 15px;
  width: 80%;
}

.exercise-item {
  display: flex;
  flex-direction: column;
  align-items: center;
  background-color: #e6e6e6;
  border-radius: 10px;
  padding: 20px;
  width: 60%;
  box-sizing: border-box;
  margin: auto;
}
```

```
.exercise-image {
  width: 100%;
  height: 70%;
  object-fit: cover;
  border-radius: 10px;
}

.exercise-details {
  text-align: center;
}

.exercise-info {
  font-size: 16px;
  margin-bottom: 5px;
}

.botons-superior {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 20px;
  margin: auto;
  border-radius: 15px;
  padding-bottom: 20px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  width: 100%;
}

.dieta-button {
  position: relative;
  width: 120%;
  /* Ancho del 80% del contenedor padre */
  max-width: 200px;
  height: 60px;
  margin-top: 50px;
  font-size: 1.5em;
  font-weight: bold;
  color: #fff;
  cursor: pointer;
  border: none;
  outline: none;
  background-size: cover;
  border-radius: 10px;
  background-image: linear-gradient(to right, #ff7300, #FFA500);
}
```

```

    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
    background-position: center;
}

.comencar-button {
    position: relative;
    width: 180%;
    /* Ancho del 80% del contenedor padre */
    max-width: 250px;
    height: 60px;
    margin-top: 20px;
    /* Reducir el espaciado superior */
    font-size: 1.5em;
    font-weight: bold;
    color: #fff;
    cursor: pointer;
    border: none;
    outline: none;
    background-size: cover;
    border-radius: 10px;
    background-image: linear-gradient(to right, #ff7300, #FFA500);
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
    background-position: center;
}

.historial-button {
    position: relative;
    width: 180%;
    /* Ancho del 80% del contenedor padre */
    max-width: 300px;
    height: 50px;
    font-size: 1.5em;
    font-weight: bold;
    color: #fff;
    cursor: pointer;
    border: none;
    outline: none;
    background-size: cover;
    border-radius: 10px;
    background-image: linear-gradient(to right, #ff7300, #FFA500);
    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);
    background-position: center;
    margin-left: 75%;
}

```

```

}

.loading {
  display: flex;
  justify-content: center;
  align-items: center;
  width: 100%;
  height: 100%;
}

.loading-image {
  width: 100px;
  /* Tamaño deseado para la imagen */
  height: auto;
  /* Mantener la proporción de aspecto */
  animation: spin 2s linear infinite;
  /* Animación de rotación */
}

@keyframes spin {
  0% {
    transform: rotate(0deg);
  }

  100% {
    transform: rotate(360deg);
  }
}

navBar {
  position: fixed;
  bottom: 0;
  width: 100%;
  z-index: 1;
}

.day-selector {
  display: flex;
  align-items: center;
  margin: auto;
}

.day-selector select {

```

```
padding: 10px;
font-size: 16px;
border-radius: 5px;
}

#rutinaBuida {
  text-align: center;
  margin-top: 20px;
  padding: 20px;
}

.buttons-container {
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 20px;
}

/* Media query para pantallas más pequeñas */
@media screen and (max-width: 790px) {
  .exercise-item {
    width: 70%;
    /* Ajuste para dos columnas en pantallas más pequeñas */
  }
}
</style>
```

- Pantalla de chat IA:

```
<style scoped>
html,
body {
  margin: 0;
  padding: 0;
  height: 100vh;
  overflow-x: hidden;
}

body {
  /* Establecer la fuente predeterminada */
```



```

background: linear-gradient(to top right, #FFA500, #f45c36);

/* Color de fondo */
}

.contenedor {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  align-items: center;
  height: 100vh;
}

.cabecera {
  background-color: #333;
  color: rgb(255, 255, 255);
  padding: 10px 0;
  font-size: 24px;
  text-align: center;
  font-weight: bold;
  border-radius: 70px;
  width: 95%;
  margin: auto;
  margin-top: 20px;
}

.mensaje-bienvenida {
  display: grid;
  grid-template-columns: .1fr 1fr;
  margin-top: 15%;
}

.botones-bienvenida{
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-gap: 20px;
  margin: auto;
  margin-top: 170px;
  margin-bottom: 20px;
  width: 90%;
}

```

```
.mensaje-bienvenida h2 {
  font-size: 1.5em;
  font-weight: 600;
  text-align: center;
  padding: 15px;

  background-color: #33333327;
  font-style: italic;
  /* Add this line to make the text italic */
  width: 70%;
  margin: auto;
  border-radius: 10px;
}

.mensaje-bienvenida img {
  width: 55px;
  height: 55px;
  border-radius: 50%;
  margin-left: 45px;
}

.chat-container {
  overflow-y: auto;
  /* Hace que el contenido sea desplazable verticalmente si es necesario */
  flex: 1;
  /* Permite que el área del chat ocupe el espacio disponible */
  width: 90%;
}

.chat {
  display: flex;
  flex-direction: column;
  margin-top: 20px;
  padding: 0 20px;
  width: 90%;
}

.chat img {
  width: 30px;
  height: 30px;
}
```

```
border-radius: 50%;
margin-right: 10px;

}

.info-usuario {
  display: grid;
  grid-template-columns: .1fr 1fr;
  align-items: center;
}

.info-usuario p {
  font-style: italic;
}

.mensaje-usuario {
  background-color: #fda65975;
  padding: 10px;
  border-radius: 25px;
  border-top-right-radius: 0;
  align-self: flex-end;
  margin-bottom: 8px;
  word-wrap: break-word;
  max-width: 90%;
}

.mensaje-asistente {
  display: flex;
  align-items: flex-start;
  margin-bottom: 8px;
  padding: 10px;
  border-radius: 25px;
  border-bottom-left-radius: 0;
  background-color: #757575a2;
  margin-right: 10%;
  max-width: 70%;
  word-wrap: break-word;
}
```

```
.avatar-asistente {
  width: 30px;
  height: 30px;
  border-radius: 50%;
  margin-right: 10px;
  background-color: #FFA500;
}

.contenido-mensaje-asistente {
  max-width: 100%;
}

.boton-chat-Rutina {
  background-color: #0000002f;
  color: white;
  border: 4px solid #1b1b1b23;
  padding: 10px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  border-radius: 4px;
  width: 80%;
  padding-top: 30px;
  padding-bottom: 30px;
  margin: auto;
}

.boton-chat-Dieta {
  background-color: #0000002f;
  color: white;
  border: 4px solid #1b1b1b23;
  padding: 10px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  border-radius: 4px;
}
```

```

width: 80%;
padding-top: 30px;
padding-bottom: 30px;
margin: auto;
}

.animacion-carga {
width: 20px;
height: 20px;
border: 2px solid #4CAF50;
border-radius: 50%;
border-top: 2px solid #ccc;
animation: spin 1s linear infinite;
align-self: flex-start;
margin-bottom: 8px;
}

@keyframes spin {
0% {
transform: rotate(0deg);
}

100% {
transform: rotate(360deg);
}
}

.controles-inferiores {
width: 100%;
display: flex;
flex-direction: column;
align-items: center;
padding-bottom: 20px;
}

.entrada-mensaje-container {
display: flex;
align-items: center;
display: grid;
grid-template-columns: 5fr .1fr .1fr;
width: 90%;
padding: 10px;

```

```
border-radius: 30px;
background-color: #333;
margin-top: 10px;
}

.entrada-mensaje {
  margin-left: 2px;
  width: 95%;
  padding-left: 10px;
  padding-top: 10px;
  box-sizing: border-box;
  background-color: #f0f0f0;
  border: none;
  border-radius: 20px;
  height: 35px;
  overflow-y: hidden;
}

.boton-enviar {
  width: 35px;
  height: 35px;
  border-radius: 50%;
  background-color: #ccc;
  color: white;
  border: none;
  display: flex;
  justify-content: center;
  align-items: center;
  cursor: pointer;
  transition: background-color 0.3s;
  margin-left: 10px;
}

.boton-enviar:hover {
  background-color: #333;
}

#send {
  width: 100%;
  height: 100%;
  margin-left: 2px;
  color: #333;
}
```

```
</style>
```

## 8. Desplegament

### Docker:

Tenim un docker que t'aixeca tota l'aplicació només fent el docker-compose up.

```
version: '3.8'

services:
  laravel:
    image: php:8.1-fpm
    container_name: Laravel
    working_dir: /var/www
    volumes:
      - ./back:/var/www
    command: >
      bash -c "apt-get update && apt-get install -y libpng-dev libjpeg-dev libfreetype6-dev zip unzip git
      libonig-dev libxml2-dev && docker-php-ext-install pdo pdo_mysql gd
      && curl -sS https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer
      && composer install
      && cp /var/www/.env.example /var/www/.env
      && php artisan key:generate
      && php artisan migrate:refresh --seed
      && php artisan serve --host=0.0.0.0 --port=8000"
    ports:
      - "8000:8000"
    depends_on:
      - db

  nuxt-app:
    image: node:18-bullseye
    container_name: Nuxt
    working_dir: /app
    volumes:
      - ./front:/app
    command: bash -c "pwd && ls -la && echo 'INI' && npm install && npm run dev"
    ports:
      - "3000:3000"
    environment:
      - CHOKIDAR_USEPOLLING=true

  db:
    image: mysql:8.2.0
    container_name: MySQL_DB
    volumes:
      - db-data2:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: root
      MYSQL_DATABASE: fithub
      MYSQL_USER: fithub
      MYSQL_PASSWORD: fithub
    ports:
      - "3307:3306"

  phpmyadmin:
    image: phpmyadmin/phpmyadmin
    container_name: PhpMyAdmin
    environment:
      PMA_HOST: db
      PMA_PORT: 3306
      PMA_USER: fithub
      PMA_PASSWORD: fithub
    ports:
      - "8080:80"
    depends_on:
      - db

volumes:
  db-data2:
```

Aixequem 4 containers, laravel, nuxt, la base de dades i el phpmyadmin per gestionarla.

- **Laravel:** En el contenidor de laravel definim el lloc de treball i li diem des d'on cap a on volem tenir-ho tot, després instal·lem totes les dependències necessàries per crear aixecar i utilitzar un laravel. Copiem el env.example a l'.env així automatitza,



no has d'anar crear tu el env cada vegada que fas un git clone. Obrim el port 8000 i depens de la base de dades

- **Nuxt:** li diem la imatge que volem, on volem treballar i de on cap a on. Després instal·lem el nuxt i el despleguem per a que puguem treballar, obrim el port 3000 i la comanda que hi ha del enviroment del nuxt es per que cada vegada que editis algo del nuxt no tinguis que anar refrescant, ho faci sol, mostri els canvis sense fer refresh.
- **BD:** li diem la imatge, el lloc de treball, d'on cap a on volem treballar, després el env per fer la connexió a la base de dades i obrim el port per poder treballar.
- **phpMyAdmin:** aquí només fa falta dir-li la imatge i nom del container, fas un environment per vincular amb el de bd els ports obrir-los i depens de bd.
- Per últim el volums de la bd

- Actions:

Utilitzem un github actions per desplegar l'aplicació més ràpid i automatizadament.

```

name: Preparant per producció
run-name: ${ github.actor } està pujant l'aplicació a PROD 🚀

on:
  push:
    branches:
      - main

jobs:
  setup-laravel-nuxt:
    runs-on: ubuntu-latest

    steps:
      - name: Obtenint el codi del repositori
        uses: actions/checkout@v2

      # Configuración de Nuxt
      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18.x'

      - name: Install npm dependencies
        run: npm install
        working-directory: ./front

      - name: Build Nuxt project
        run: npm run build
        working-directory: ./front

      # Generate Nuxt project
      - name: Generate Nuxt static files
        run: npm run generate
        working-directory: ./front

      # Verificación de la generación de Nuxt
      - name: Verificar la generación de Nuxt
        run: ls -la ./front/dist
        working-directory: ./

      # Exportar el secret como variable de entorno
      - name: Set API key as environment variable
        run: echo "OPENAI_API_KEY=${ secrets.OPENAI_API_KEY }}" >> $GITHUB_ENV

      # Subida a producción
      - name: Subir el build de Nuxt y la carpeta back a producción con SCP
        run: |
          echo "${ secrets.PROD_KEY }}" > ~/prod_key.pem
          chmod 600 ~/prod_key.pem
          scp -r -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i "~/prod_key.pem" ./front/dist/*
            ${ secrets.PROD_USER }}@${ secrets.PROD_HOST }:/home/a20aitbaresc/web/fithub.daw.inspedralbes.cat/public_html/
          scp -r -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null -i "~/prod_key.pem" ./back/*
            ${ secrets.PROD_USER }}@${ secrets.PROD_HOST }:/home/a20aitbaresc/web/fithub.daw.inspedralbes.cat/public_html/

      - run: echo "💖 This job's status is ${ job.status }."

```

Lo que fa aquest actions es quan fas commit a main, agafa una ubuntu més recent, agafa el codi de la nostra aplicació, agafa l'última versió del node, instal·la les dependències del nuxt, i li dius que el lloc de treball es ./front, després fa un build, després fa un generate, comprova que el generate este ben fet, exportem el secret de la api key per que no digui res el github, i pujem a producció el back i el contingut del dist a public\_html de la nostra url.