Delishare

Cook. Share. Inspire.

Document tecnic

Simran Kaur
Ishaa Amin
2DAW

ÍNDEX

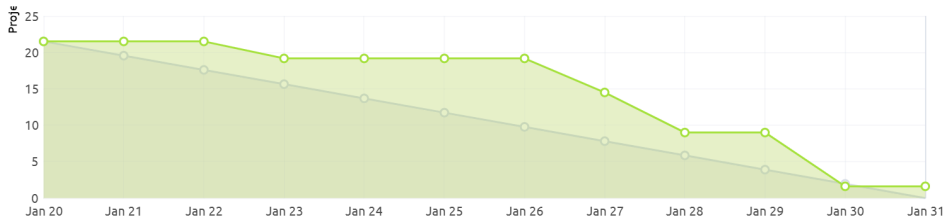
DeliShare
RECIPE WEBSITE

Evolució de les funcionalitats al llarg dels Sprints

Sprint 1 – Bases del projecte

En aquest primer sprint ens vam centrar en establir l'estructura bàsica del projecte. Vam crear totes les **migracions**, el sistema de **registre i inici de sessió**, i vam preparar la **landing page** per mostrar receptes.

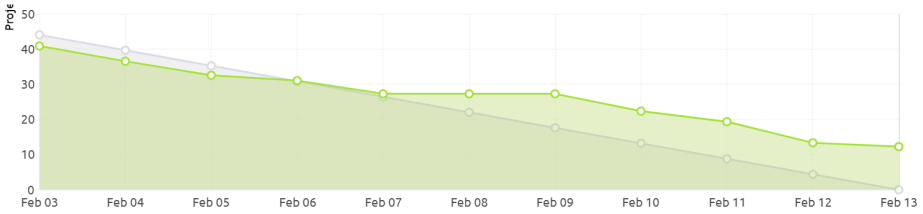
També vam implementar la **pujada d'imatges** amb **Cloudinary** a l'hora de crear receptes, i vam desenvolupar la **pàgina de cerca** amb filtres per facilitar la navegació entre receptes.



Sprint 2 – Publicació i primer contacte amb la IA

En aquest sprint vam pujar el projecte a **producció** per començar a provar-lo en un entorn real. També vam començar a treballar la part de **IA**, perquè al crear una recepta es poguessin **autocompletar els passos** i afegir informació extra de manera automàtica.

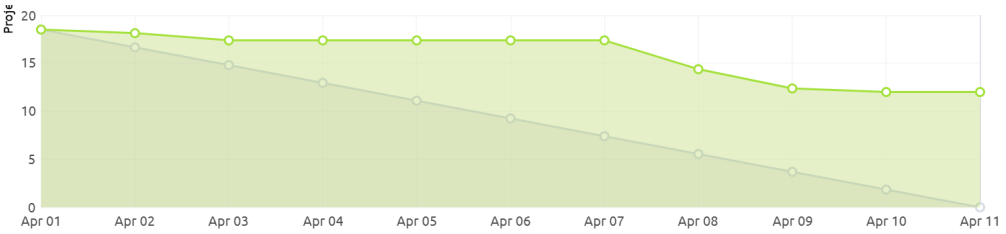
A més, vam implementar la funcionalitat per **pujar o canviar la foto de perfil**, i vam afegir la possibilitat de veure **informació d'altres usuaris** des del seu perfil públic.



Sprint 3 – Millores en la seguretat i noves funcionalitats

Durant aquest sprint vam treballar en **restriccions amb la IA** per controlar millor el moment de crear receptes i pujar imatges, assegurant-nos que es compleixin certs requisits. També vam implementar que, sense haver fet **login**, no es poguessin veure els **detalls d'una recepta**.

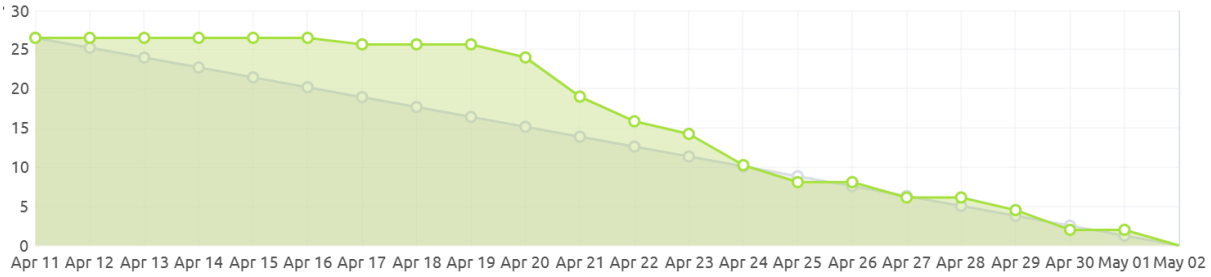
Vam afegir la funcionalitat perquè, al fer el **registre**, l'usuari rebés un **correu de confirmació**. A més, vam permetre **descarregar informació sobre les receptes** i vam crear un apartat de **notificacions** per mantenir a l'usuari al dia. Finalment, vam implementar una secció per veure les **receptes a les que l'usuari ha donat like**.



Sprint 4 – Disseny i funcionalitats avançades

En aquest últim sprint, vam treballar per fer que el projecte fos **responsive** i tingués un **disseny atractiu** i intuïtiu per a tots els dispositius. També vam implementar un **CRUD d'administrador** per gestionar receptes, usuaris i comentaris.

Vam afegir una funcionalitat per a la **nutrició de les receptes**, que es recalcula automàticament quan es modifiquen les quantitats d'ingredients. Finalment, vam integrar un sistema de **temps real** per millorar l'experiència de l'usuari, com el xat o actualitzacions instantànies.



Sprint 5 – Millores en la cerca, rols i disseny

En aquest sprint vam implementar la funcionalitat perquè es pogués **buscar un live per nom o per xef**, i que al **perfil d'un usuari** es veiessin també els seus **lives**. A més, vam permetre que un **usuari pogués sol·licitar un canvi de rol** enviant un correu a l'adreça **administracio@delishare.cat**, i vam donar als administradors la possibilitat de **canviar el rol d'un usuari** des del backend.

També vam treballar en el **disseny** del projecte, utilitzant **Tailwind CSS** per aconseguir una aparença més moderna i responsive. Finalment, vam **pujar el projecte a producció**, posant en marxa totes les funcionalitats noves.

DeliShare
RECIPE WEBSITE

Funcionalitats Noves

1. Càmera i àudio en temps real amb WebSockets

Hem afegit la funcionalitat de transmissió de vídeo i àudio mitjançant WebSockets, la qual cosa permet als usuaris emetre en directe o participar en "lives" amb altres persones. Aquesta funcionalitat aprofita l'accés a la càmera i al micròfon del dispositiu, permetent comunicació audiovisual en temps real dins de la plataforma.

2. Intel·ligència Artificial per validar imatges i vídeos

Hem implementat un sistema basat en IA que revisa automàticament les imatges i vídeos que pugen els usuaris. Aquesta intel·ligència detecta contingut inadequat o no relacionat amb la cuina, i bloqueja la pujada si no compleix amb els criteris definits (com per exemple, contingut ofensiu, imatges sexuals o vídeos no culinaris).

3. Disseny amb Tailwind CSS

Per millorar la interfície i facilitar la mantenibilitat del codi, hem utilitzat **Tailwind CSS**, un framework modern que permet dissenyar de forma ràpida i coherent. Això ens ha permès tenir un disseny responsive, modern i accessible a tots els dispositius, amb una estètica uniforme i professional.

4. Enviament de correus i notificacions des del domini de Delishare

Ara els correus electrònics (com la confirmació de registre, notificacions de receptes o canvis de rol) es fan a través del nostre propi domini personalitzat, **@delishare.cat**, utilitzant el servei de **Nominalia**.
Això afegeix un toc de professionalitat i fiabilitat, ja que els usuaris reben correus directament des de la marca del projecte.

Problemes i solucions

Problema amb la pujada d'imatges de perfil

Un dels problemes que vam tenir va ser no saber com permetre als usuaris pujar i veure la seva imatge de perfil. Després de buscar solucions, vam decidir utilitzar **Cloudinary**, una eina al núvol per gestionar imatges.

Des del frontend amb Vue pugem la imatge a Cloudinary, i des del backend amb Laravel guardem la URL a la base de dades. Així podem mostrar fàcilment la imatge al perfil de cada usuari.

Restriccions en la pujada de fitxers

També vam tenir problemes a l'hora d'aplicar restriccions específiques quan es pujava una imatge o vídeo. No sabíem com fer-ho exactament perquè es complissin les condicions que volíem.

Vam demanar ajuda a una companya de DAM, que ens va explicar que a la IA (o a qualsevol sistema) cal indicar-li de forma molt precisa què volem i, sobretot, què **no** volem. Així ho vam fer: vam definir clarament els límits i condicions, com el pes màxim del fitxer o els formats acceptats, i finalment ho vam poder implementar correctament.

Implementació de WebSockets i funcionalitat en directe

Un altre repte va ser no saber com implementar WebSockets al projecte. Volíem fer una funcionalitat en temps real entre usuàries, però no teníem clar per on començar.

Després de demanar ajuda als professors, vam decidir començar per una base més senzilla: implementar primer un xat en temps real. Un cop això va funcionar correctament, vam continuar millorant-lo per oferir una millor experiència d'usuari.

DeliShare
RECIPE WEBSITE

Generació Automàtica de Receptes amb Groq AI

Introducció

Aquest mòdul utilitza l'API de **Groq AI** per generar receptes de cuina completes de manera automàtica, basant-se en les preferències de l'usuari. La implementació combina intel·ligència artificial amb una interfície amigable per simplificar el procés de creació de receptes.

Flux de Generació Automàtica

1. Interacció amb Groq AI

El sistema utilitza el model llama-3.3-70b-versatile de **Groq** per generar receptes completes. Quan l'usuari fa clic al botó **"Omplir automàticament"**, es produeix el següent:

- Es recopilen les dades introduïdes per l'usuari (títol, explicació, categoria, tipus de cuina i racions)
- Es construeix un prompt detallat amb instruccions estrictes per garantir que la resposta sigui exclusivament relacionada amb cuina
- S'envia la sol·licitud a l'API de Groq amb el format específic requerit
- La resposta s'analitza i s'integra automàticament al formulari

2. Validació de Contingut

Per garantir la qualitat i la seguretat del contingut generat:

- S'aplica un filtre de paraules prohibides que rebutja contingut inadequat
- Es valida que la resposta de la IA sigui un **JSON** vàlid amb l'estructura esperada
- Es verifica que tots els camps essencials estiguin presents (ingredients, passos, informació nutricional)

3. Components Generats

La IA genera automàticament:

- Una descripció breu però informativa
- Una llista completa d'ingredients amb quantitats exactes
- Passos detallats de preparació

- Valors nutricionals aproximats per ració
- Temps estimats de preparació i cocció

Càlcul Nutricional Automàtic

1. Funcionament

El sistema inclou un **calculador nutricional intel·ligent** que:

- Analitza els ingredients introduïts per l'usuari
- Utilitza Groq AI per estimar els valors nutricionals
- Actualitza automàticament els camps de calories, proteïnes, greixos i carbohidrats
- Té en compte el nombre de racions per calcular els valors per porció

2. Optimització

El càlcul es realitza amb **debounce** (retard de 1,5 segons) per:

- Evitar crides innecessàries a l'API
- Reduir costos operatius
- Millorar l'experiència d'usuari durant l'edició

Beneficis Clau

1. **Estalvi de temps**: Genera receptes completes en pocs segons
2. **Inspiració culinària**: Ajuda els usuaris amb bloqueig creatiu
3. **Precisió nutricional**: Proporciona estimacions basades en els ingredients
4. **Personalització**: Adapta les receptes a categories i tipus de cuina específics
5. **Control de qualitat**: Filtra contingut inadequat i valida que sigui sobre cuina

---

Consideracions de Seguretat

- Validació estricta del contingut generat
- Filtratge de paraules inapropiades
- Restricció per generar només contingut culinari
- Protecció contra possibles atacs per injecció de prompts

Prompt

` ESTRICTE: Només pots generar receptes de cuina. Rebutja absolutament qualsevol altre tipus de sol·licitud.

Genera UNA RECEPTA DE CUINA en format JSON basada en aquestes dades:
- Categoria: ${categoryName}
- Cuina: ${cuisineName}
- Racions: ${servings}
- Títol: ${recipe.value.title || "[crea un títol atractiu]"}
- Explicació del que vol fer l'usuari: ${recipe.value.explanation}

La recepta ha de ser EXCLUSIVAMENT sobre cuina i ha d'incloure:
1. Descripció breu (màxim 2 frases)
2. Llista d'ingredients amb quantitats exactes
3. Passos de preparació numerats
4. Informació nutricional per ració
5. Temps de preparació i cocció

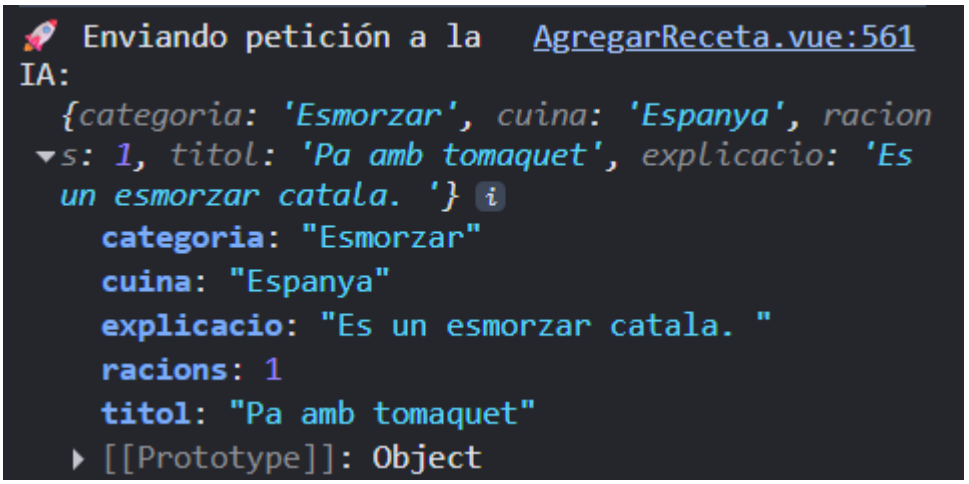
FORMAT REQUERIT (en català, només JSON):

```
{
"title": "Títol de la recepta",
"description": "Descripció breu",
"ingredients": [
  {"quantity": "quantitat exacta", "unit": "unitat (obligatori)", "name": "Nom ingredient"},
    ...
],
"steps": [
    "Descripció detallada de pas 1",
    "Descripció detallada de pas 2",
    ...
],
"nutrition": {
    "calories": 0,
    "protein": 0,
    "fats": 0,
    "carbs": 0
},
"times": {
```

```
    "prep_time": 0,
    "cook_time": 0
}
}
```

NORMES ESTRICTES:
- ABSOLUTAMENT RES que no sigui sobre cuina/receptes
- No inclour cap tipus de contingut inadequat
- Quantitats precises i reals
- Passos clars i executables
- Informació nutricional realista
- Resposta EXCLUSIVAMENT en format JSON vàlid `

Datos enviados a la ia-



Resposta obtinguda-

DeliShare
RECIPE WEBSITE

Validació d’Imatges i Vídeos amb Gemini

Aquest servei permet validar si una imatge o un vídeo conté menjar, utilitzant el model Gemini 1.5 Flash de Google a través de la seva API oficial (@google/generative-ai).

Validació d’Imatges

El procés de validació d’imatges es basa a convertir el fitxer a base64 i enviar-lo al model Gemini juntament amb un prompt redactat en català. Aquest prompt sol·licita al model que actuï com un expert en gastronomia i analitzi si la imatge mostra menjar, ingredients, plats cuinats o persones menjant. La resposta esperada és un JSON amb dos camps: isFood (booleà que indica si hi ha menjar o no) i reason (explicació en català).

Prompt clau :

`Eres un experto en análisis de imágenes de comida. Tu tarea es determinar si la imagen proporcionada contiene comida o no.

Devuelve un JSON con el siguiente formato:

{

"isFood": boolean,

"reason": "string" }

Reglas:

- Si la imagen muestra claramente comida (platos preparados, ingredientes, etc.), marca isFood como true

- Si la imagen muestra personas pero están comiendo o con comida, marca isFood como true

- Si la imagen no contiene comida (personas solas, paisajes, objetos, etc.), marca isFood como false

- Si no puedes determinar con seguridad, marca isFood como false`

Validació de Vídeos

La validació de vídeos es realitza extraient diversos fotogrames en moments específics (1, 5, 10, 15 i 20 segons). Cada fotograma es converteix a base64 i s’analitza individualment amb el mateix procés que en les imatges. Si almenys un dels fotogrames conté menjar, el vídeo es considera vàlid. El resultat també es retorna en format JSON amb un camp isFood i una reason explicant el veredicte.

Prompt clau:

`Ets un expert en vídeos de cuina. Aquesta imatge és un fotograma extret d'un vídeo. La teva tasca és determinar si mostra aliments o escenes relacionades amb la cuina.

Retorna un JSON amb el format següent:

{

"isFood": boolean,

"reason": "string"}

Recorda:

- Si el fotograma mostra clarament menjar, ingredients o cuina, és vàlid.

- Si no sembla rellevant a la cuina, és invàlid.`

Output esperat:

Tant per imatges com per vídeos:

{

"isFood": true | false,

"reason": "explicació en català"

}

Consideracions

- Totes les anàlisis es fan en català, tant en els prompts com en les respostes del model.
- S'utilitzen funcions auxiliars per a la conversió de fitxers (fileToBase64) i l'extracció de fotogrames de vídeo (extractFramesFromVideo).
- La validació de vídeos es fa de manera paral·lela per optimitzar el rendiment.

Aquest sistema permet realitzar una moderació intel·ligent de contingut visual, filtrant material que no estigui relacionat amb menjar abans de ser publicat a la plataforma.

DeliShare
RECIPE WEBSITE

Rutas api

Autenticació i Usuari:

- POST /register: registra un usuari nou amb el controlador AuthController, funció register.
- POST /login: inicia sessió amb AuthController, funció login.
- GET /userInfo/{userId}: obté informació de l'usuari amb AuthController, funció getUserInfo.
- POST /logout: tanca sessió amb AuthController, funció logout.
- POST /updatePerfile: actualitza el perfil amb AuthController, funció updatePerfil.
- PUT /updateProfilePicture: actualitza la imatge de perfil amb AuthController, funció updateProfilePicture.
- POST /cambiarContra: canvia la contrasenya amb AuthController, funció cambiarContra.
- PUT /usuarios/{id}/rol: canvia el rol d'un usuari amb AuthController, funció cambiarRol.

Usuaris:

- GET /users: llista tots els usuaris amb UserController, funció index.
- GET /users/{id}: mostra un usuari específic amb UserController, funció show.
- DELETE /users/{id}: elimina un usuari amb UserController, funció destroy.
- POST /users/: crea un usuari amb UserController, funció store.
- GET /getAllUsers: obté tots els usuaris des de RecipeController, funció getAllUsers.
- POST /send-verification: envia correu de verificació amb UserController, funció sendVerificationEmail.

Receptes:

- GET /recipes: llista receptes amb RecipeController, funció index.
- GET /recipes/{id}: mostra una recepta amb RecipeController, funció show.
- POST /recipes: crea una recepta amb RecipeController, funció store.
- PUT /recipes/{id}: actualitza una recepta amb RecipeController, funció update.

- DELETE /recipes/{id}: elimina una recepta amb RecipeController, funció destroy.
- GET /getAllRecipes: obté totes les receptes amb RecipeController, funció getAllRecipes.
- GET /user/recetas: obté receptes de l'usuari amb RecipeController, funció getRecipesByUser.
- GET /recipes/{id}/steps: obté els passos d'una recepta amb RecipeController, funció getRecipeSteps.
- GET /recipes/{id}/download: permet descarregar la recepta completa amb RecipeController, funció downloadFullRecipe.
- GET /ingredients: obté tots els ingredients amb RecipeController, funció getAllIngredients.
- POST /recipes/filter-by-ingredients: filtra per ingredients amb RecipeController, funció filterByIngredients.

Categories:

- POST /categories: crea una categoria amb CategoryController, funció store.
- GET /categories: llista totes les categories amb CategoryController, funció index.
- GET /categories/{id}: mostra una categoria amb CategoryController, funció show.
- PUT /categories/{id}: actualitza una categoria amb CategoryController, funció update.
- DELETE /categories/{id}: elimina una categoria amb CategoryController, funció destroy.
- GET /filterByCategory/{id}: filtra receptes per categoria amb RecipeController, funció filterByCategory.

Cuines:

- POST /cuisines: crea una cuina amb CuisineController, funció store.
- GET /cuisines: llista totes les cuines amb CuisineController, funció index.
- GET /cuisines/{id}: mostra una cuina amb CuisineController, funció show.
- PUT /cuisines/{id}: actualitza una cuina amb CuisineController, funció update.
- DELETE /cuisines/{id}: elimina una cuina amb CuisineController, funció destroy.
- GET /filterByCuisine/{id}: filtra receptes per cuina amb CuisineController, funció filterByCuisine.

Comentaris:

- POST /recipes/{id}/comment: afegeix un comentari a una recepta amb RecipeController, funció addComment.
- GET /recipes/{id}/comments: obté comentaris d'una recepta amb RecipeController, funció getRecipeComments.
- DELETE /recipes/{recipeId}/comments: elimina un comentari pel seu text amb RecipeController, funció deleteCommentByText.
- GET /comments: obté tots els comentaris amb RecipeController, funció getAllComments.

M'agrada i Guardats:

- POST /recipes/{recipe}/like: alterna el "m'agrada" amb RecipeController, funció toggleLike.
- GET /recipes/{recipe}/likes: obté els "m'agrada" d'una recepta amb RecipeController, funció getLikes.
- POST /recipes/{id}/like: marca com a "m'agrada" una recepta amb RecipeController, funció likeRecipe.
- POST /recipes/{id}/unlike: treu el "m'agrada" amb RecipeController, funció unlikeRecipe.
- GET /saved-recipes: llista receptes guardades amb SavedRecipeController, funció index.
- POST /saved-recipes/toggle/{recipeId}: guarda o treu una recepta guardada amb SavedRecipeController, funció toggleSave.

Carpetes:

- POST /folders: crea una carpeta amb FolderController, funció store.
- POST /folders/{folderId}/recipes/{recipeId}: afegeix una recepta a una carpeta amb FolderController, funció addRecipe.
- GET /folders/{folder}: mostra una carpeta amb FolderController, funció show.
- DELETE /folders/{folderId}/recipes/{recipeId}: elimina una recepta d'una carpeta amb FolderController, funció removeRecipe.
- DELETE /folders/{folderId}: elimina una carpeta amb FolderController, funció removeFolder.
- GET /folders: llista totes les carpetes amb FolderController, funció index.
- GET /folders/{folder}/recipes: obté receptes dins d'una carpeta amb FolderController, funció getRecipes.

Lives:

- GET /lives: llista transmissions en viu amb LiveController, funció index.
- POST /lives: crea una transmissió amb LiveController, funció store.
- GET /lives/{live}: mostra una transmissió amb LiveController, funció show.
- PUT /lives/{live}: actualitza una transmissió amb LiveController, funció update.
- DELETE /lives/{live}: elimina una transmissió amb LiveController, funció destroy.
- GET /mis-lives: obté les meves transmissions amb LiveController, funció chefLives.
- GET /users/{userId}/lives: obté transmissions d'un usuari amb LiveController, funció getUserLives.

Notificacions:

- GET /notifications: obté notificacions de l'usuari amb NotificationController, funció getUserNotifications.
- POST /notifications: crea una notificació amb NotificationController, funció createNotification.
- PUT /notifications/{id}/read: marca una notificació com a llegida amb NotificationController, funció markAsRead.

Recomanacions:

- GET /recommendations/preferences: obté preferències de l'usuari amb RecommendationController, funció getUserPreferences.
- POST /recommendations/preferences: desa preferències amb RecommendationController, funció storePreferences.
- GET /user/preferences: obté noms de preferències amb RecommendationController, funció getPreferenceNames.
- GET /recipes/recommended: obté receptes recomanades amb RecipeController, funció getRecommendedRecipes.
- GET /recommendations/options: obté cuines i categories amb RecommendationController, funció getCuisinesAndCategories.
- GET /user/liked-recipes: obté receptes marcades com a "m'agrada" per l'usuari amb RecipeController, funció getUserLikedRecipes.

Filtres:

- GET /filterByTime/{time}: filtra receptes per temps amb RecipeController, funció filterByTime.
- GET /times: obté totes les opcions de temps amb RecipeController, funció getAllTimes.

Nota: Totes les rutes estan protegides pel middleware auth:sanctum, excepte aquelles que són públiques explícitament, com el registre, el login i algunes de visualització.