

Requisiti

Domanda : che requisiti dovrebbe avere una app per aiutare una alimentazione sostenibile

Risposta: Per aiutare a promuovere un'alimentazione sostenibile, un'app dovrebbe soddisfare una serie di requisiti e funzionalità che incoraggino pratiche alimentari ecologiche e consapevoli. Ecco alcune caratteristiche chiave:

1. Educazione e Informazione:

- **Informazioni Nutrizionali e Ambientali:** Fornire dettagli sull'impatto ambientale degli alimenti (ad esempio, impronta di carbonio, uso dell'acqua).
- **Guide e Risorse:** Offrire articoli, video e tutorial su argomenti come l'agricoltura biologica, l'alimentazione a base vegetale, e le pratiche di riduzione degli sprechi alimentari.

2. Tracciabilità degli Alimenti:

- **Origine dei Prodotti:** Permettere agli utenti di tracciare l'origine dei loro alimenti per garantire che provengano da fonti sostenibili.
- **Certificazioni e Etichette:** Fornire informazioni su certificazioni di sostenibilità come Fair Trade, Rainforest Alliance, e certificazioni biologiche.

3. Pianificazione dei Pasti:

- **Ricette Sostenibili:** Offrire ricette che utilizzano ingredienti di stagione e locali, riducendo così l'impatto ambientale.
- **Calcolo delle Porzioni:** Aiutare gli utenti a pianificare le porzioni corrette per ridurre gli sprechi alimentari.

4. Gestione degli Acquisti:

- **Liste della Spesa Intelligenti:** Generare liste della spesa basate sulle ricette pianificate, favorendo l'acquisto di prodotti freschi e sostenibili.
- **Suggerimenti per l'Acquisto di Prodotti Locali:** Indicare mercati locali e negozi che vendono prodotti sostenibili.

5. Riduzione degli Sprechi:

- **Conservazione degli Alimenti:** Fornire consigli su come conservare correttamente gli alimenti per prolungarne la durata.
- **Ricette con Avanzi:** Offrire ricette che utilizzano avanzi o cibi prossimi alla scadenza.

6. Community e Condivisione:

- **Forum e Social:** Creare uno spazio per gli utenti dove possono condividere ricette, consigli e esperienze legate all'alimentazione sostenibile.
- **Sfide e Obiettivi:** Proporre sfide settimanali o mensili che incentivino pratiche sostenibili, come ridurre il consumo di carne o utilizzare solo prodotti di stagione.

7. Personalizzazione e Monitoraggio:

- **Profilo Utente:** Permettere agli utenti di impostare preferenze alimentari e obiettivi di sostenibilità personalizzati.
- **Monitoraggio dei Progressi:** Tenere traccia dei progressi dell'utente verso i suoi obiettivi di sostenibilità alimentare e fornire feedback e suggerimenti per miglioramenti ulteriori.

8. Accessibilità e Usabilità:

- **Interfaccia Intuitiva:** Garantire che l'app sia facile da usare con un'interfaccia utente chiara e accessibile.
- **Supporto Multilingue:** Offrire supporto per diverse lingue per raggiungere un pubblico più ampio.

9. Integrazione con Altre Piattaforme:

- **Connessione con Servizi di Consegna di Cibo:** Integrare con servizi che offrono cibo sostenibile e locale.
- **Sincronizzazione con App di Fitness e Salute:** Collegarsi con altre app di salute e benessere per fornire una visione olistica delle abitudini alimentari dell'utente.

Implementando questi requisiti, un'app può giocare un ruolo significativo nel promuovere abitudini alimentari più sostenibili, educando gli utenti e facilitando scelte più consapevoli e responsabili.

PARZIALMENTE UTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: come creo una classe viewModel factory in java

RISPOSTA: In Java, una classe `ViewModelFactory` viene utilizzata in combinazione con l'architettura MVVM (Model-View-ViewModel) per istanziare e fornire istanze dei `ViewModel` alle classi di `View`. Ecco un esempio di come potresti creare una classe `ViewModelFactory` in Java:

```
java
Copia codice
import androidx.annotation.NonNull;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProvider;

public class ViewModelFactory implements ViewModelProvider.Factory {

    // Qualsiasi dipendenza che potrebbe essere necessaria per creare i
    ViewModels
    private final SomeRepository someRepository;

    public ViewModelFactory(SomeRepository someRepository) {
        this.someRepository = someRepository;
    }

    @NonNull
    @Override
    public <T extends ViewModel> T create(@NonNull Class<T> modelClass) {
        if (modelClass.isAssignableFrom(SomeViewModel.class)) {
            // Se la classe del ViewModel è SomeViewModel, istanzia e
            restituisci un'istanza di SomeViewModel
            return (T) new SomeViewModel(someRepository);
        }
        // Aggiungi altri controlli per altre classi di ViewModel se necessario
        throw new IllegalArgumentException("Unknown ViewModel class");
    }
}
```

In questo esempio:

- `ViewModelFactory` implementa l'interfaccia `ViewModelProvider.Factory`, che richiede un metodo `create` per creare istanze di `ViewModel`.
- Nel costruttore della classe `ViewModelFactory`, puoi passare qualsiasi dipendenza necessaria per creare i `ViewModels`. Nel caso sopra, `SomeRepository` è passato come dipendenza.
- Nel metodo `create`, si controlla la classe del `ViewModel` richiesta (`modelClass`). Se è una classe di `ViewModel` nota (nel nostro esempio, `SomeViewModel`), viene istanziata e restituita un'istanza di quel `ViewModel`. Altrimenti, viene lanciata un'eccezione `IllegalArgumentException` per indicare che la classe richiesta non è gestita.

Assicurati di adattare questa implementazione alle tue esigenze specifiche, aggiungendo eventuali altri controlli o dipendenze necessarie per i `ViewModels` della tua applicazione.

UTILE

RIELABORAZIONE MARGINALE

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: e come dovrebbe essere il viewmodel associato

RISPOSTA: Ecco un esempio di come potrebbe essere implementato il ViewModel associato, ad esempio `SomeViewModel`:

```
java
Copia codice
import androidx.lifecycle.ViewModel;

public class SomeViewModel extends ViewModel {

    private final SomeRepository someRepository;
    // Altri campi e metodi del ViewModel, se necessario

    public SomeViewModel(SomeRepository someRepository) {
        this.someRepository = someRepository;
        // Altri inizializzazioni, se necessario
    }

    // Metodi per interagire con il repository o altre logiche del ViewModel

    // Esempio di un metodo per ottenere dati dal repository
    public LiveData<SomeData> getSomeData() {
        return someRepository.getSomeData();
    }
}
```

In questo esempio:

- `SomeViewModel` estende la classe `ViewModel` fornita da Android Jetpack. Tutta la logica relativa alla gestione dei dati e delle operazioni di business dovrebbe essere inclusa all'interno di questa classe.
- Viene accettato un riferimento a `SomeRepository` nel costruttore per consentire al `ViewModel` di accedere ai dati.
- Il `ViewModel` espone metodi che le classi di `View` possono chiamare per ottenere dati o per effettuare altre azioni. In questo caso, abbiamo un esempio di metodo `getSomeData()` che potrebbe essere utilizzato per ottenere dei dati da `SomeRepository`.

Assicurati di adattare `SomeViewModel` alle necessità specifiche della tua applicazione, aggiungendo eventuali altri metodi o campi necessari per supportare le funzionalità della tua `View`.

RISPOSTA UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: come dovrei chiamare un metodo di una repository da android

RISPOSTA: Per chiamare un metodo di una repository da un'applicazione Android, segui questi passaggi:

1. **Importa la repository nel tuo progetto Android:** Assicurati di avere importato correttamente la repository nel tuo progetto Android. Questo può includere l'aggiunta della dipendenza nel file `build.gradle` del tuo modulo o qualsiasi altro metodo che utilizzi per gestire le dipendenze.
2. **Inizializza l'istanza della repository:** Crea un'istanza della repository nel tuo codice, utilizzando il costruttore appropriato o un metodo di inizializzazione.
3. **Chiamata del metodo desiderato:** Una volta che hai un'istanza della tua repository, puoi chiamare i suoi metodi normalmente.

Ecco un esempio di come potresti fare questo in Android:

```
java
Copia codice
// Supponiamo che tu abbia una classe UserRepository che implementa un metodo
getUserById()
UserRepository userRepository = new UserRepository();

// Ora puoi chiamare il metodo getUserById() passando l'id dell'utente
User user = userRepository.getUserById(userId);
```

Assicurati di gestire eventuali eccezioni che possono essere sollevate durante questa chiamata, ad esempio le eccezioni di rete o di database, a seconda di come è implementata la tua repository.

RISPOSTA PARZIALMENTE UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: come faccio a leggere un json ricevuto da chiamate api su android

RISPOSTA: Per leggere un JSON ricevuto da chiamate API su Android, puoi seguire questi passaggi:

1. **Effettua la chiamata API:** Prima di tutto, effettua la chiamata API per ottenere il JSON desiderato. Puoi utilizzare librerie come Retrofit, Volley o HttpClient per effettuare la chiamata API. Assicurati di gestire i permessi di rete nel tuo manifest, se necessario.
2. **Parsa il JSON:** Una volta ricevuto il JSON, devi parsarlo per estrarre i dati che ti interessano. Puoi farlo utilizzando le classi `JSONObject` e `JSONArray` in Java.

Ecco un esempio di come puoi fare questo:

```
java
Copia codice
// Supponiamo che "response" sia la risposta JSON ricevuta dalla chiamata API

try {
    JSONObject jsonResponse = new JSONObject(response); // Converte la risposta
in un oggetto JSON

    // Estrai i dati dal JSON
    String name = jsonResponse.getString("name");
    int age = jsonResponse.getInt("age");

    JSONArray jsonArray = jsonResponse.getJSONArray("items"); // Se il JSON
contiene un array

    for (int i = 0; i < jsonArray.length(); i++) {
        JSONObject item = jsonArray.getJSONObject(i);
        // Esegui l'elaborazione per ogni elemento dell'array
    }

} catch (JSONException e) {
    e.printStackTrace();
}
```

Assicurati di gestire adeguatamente le eccezioni in caso di errori di parsing o altri problemi con la risposta JSON. Puoi usare blocchi `try-catch` come mostrato nell'esempio sopra.

3. **Gestisci i dati:** Dopo aver estratto i dati dal JSON, puoi utilizzarli nel tuo codice Android come necessario, ad esempio per aggiornare l'interfaccia utente o salvare i dati nel database locale. Assicurati di eseguire le operazioni di parsing e gestione dei dati in un thread separato rispetto al thread dell'interfaccia utente per evitare il blocco dell'applicazione. Puoi utilizzare `AsyncTask`, `Handler` o librerie come `RxJava` per questo scopo.

INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGERATA

FIXING

DOMANDA: nvece per un gson

RISPOSTA: Utilizzare la libreria Gson semplifica notevolmente il parsing dei dati JSON in oggetti Java su Android. Ecco come puoi farlo:

1. **Aggiungi la dipendenza Gson:** Assicurati di aver aggiunto la dipendenza Gson nel tuo file `build.gradle` del modulo:

```
groovy
Copia codice
implementation 'com.google.code.gson:gson:2.8.8'
```

2. **Effettua la chiamata API e ottieni la risposta JSON:** Utilizza Retrofit, Volley o qualsiasi altra libreria di tua scelta per effettuare la chiamata API e ottenere la risposta JSON.
3. **Esegui il parsing del JSON usando Gson:** Utilizza Gson per convertire la risposta JSON in oggetti Java.

Ecco un esempio di come puoi fare questo:

```
java
Copia codice
// Supponiamo che "jsonResponse" sia la risposta JSON ricevuta dalla chiamata API

// Crea un oggetto Gson
Gson gson = new Gson();

try {
    // Parsa la risposta JSON in un oggetto Java corrispondente
    YourObject yourObject = gson.fromJson(jsonResponse, YourObject.class);

    // Ora puoi utilizzare l'oggetto yourObject come desiderato
    String name = yourObject.getName();
    int age = yourObject.getAge();
} catch (JsonSyntaxException e) {
    e.printStackTrace();
}
```

Assicurati di sostituire "YourObject" con la classe Java corrispondente che rappresenta la struttura del JSON. Gson converte automaticamente la struttura del JSON nei campi corrispondenti della classe Java. Se il JSON contiene un array di oggetti, puoi utilizzare il metodo `fromJson()` per convertire l'array JSON in una lista di oggetti Java.

```
java
Copia codice
YourObject[] objectsArray = gson.fromJson(jsonArrayResponse,
YourObject[].class);
```

Assicurati di gestire adeguatamente le eccezioni, come `JsonSyntaxException`, in caso di errori di parsing.

INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: come inizializzare una classe viewModel

RISPOSTA: Per inizializzare una classe ViewModel in un'applicazione Android, segui questi passaggi:

1. **Aggiungi la dipendenza ViewModel:** Assicurati di avere aggiunto la dipendenza ViewModel nel tuo file `build.gradle` del modulo:

```
groovy
Copia codice
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.0"
```

2. **Crea la tua classe ViewModel:** Crea una classe che estenda `ViewModel` e aggiungi la logica necessaria al suo interno.

```
java
Copia codice
import androidx.lifecycle.ViewModel;

public class MyViewModel extends ViewModel {
    // Aggiungi qui i tuoi campi dati e metodi
}
```

3. **Inizializza la classe ViewModel nel tuo codice:** Inizializza la tua classe `ViewModel` all'interno di un'attività (`Activity`) o di un frammento (`Fragment`). Puoi farlo utilizzando la classe `ViewModelProvider`.

```
java
Copia codice
import androidx.lifecycle.ViewModelProvider;

// Dentro un'Activity o Fragment

// Inizializza la tua classe ViewModel
MyViewModel viewModel = new ViewModelProvider(this).get(MyViewModel.class);
```

Se stai inizializzando la `ViewModel` all'interno di un'Activity, puoi utilizzare `this` come primo argomento per `ViewModelProvider`. Se stai inizializzando la `ViewModel` all'interno di un `Fragment`, puoi utilizzare `requireActivity()` come primo argomento.

4. **Utilizza la ViewModel:** Ora puoi utilizzare l'oggetto `viewModel` per accedere ai campi dati e ai metodi definiti nella tua classe `ViewModel`.

```
java
Copia codice
// Esempio di utilizzo
viewModel.doSomething();
```

La classe `ViewModelProvider` si occupa di gestire il ciclo di vita della tua `ViewModel` e garantirà che la stessa istanza della `ViewModel` venga fornita all'attività o al frammento anche durante i cambiamenti di configurazione, come le rotazioni dello schermo.

RISPOSTA UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: utilizzando una factory

RISPOSTA: Per inizializzare una classe `ViewModel` utilizzando una factory, segui questi passaggi:

1. **Definisci una factory per la `ViewModel`:** Crea una classe che implementi l'interfaccia `ViewModelProvider.Factory`. Questa factory sarà responsabile di creare e fornire un'istanza della `ViewModel` desiderata.

```
java
Copia codice
import androidx.annotation.NonNull;
import androidx.lifecycle.ViewModel;
import androidx.lifecycle.ViewModelProvider;

public class MyViewModelFactory implements ViewModelProvider.Factory {

    @NonNull
    @Override
    public <T extends ViewModel> T create(@NonNull Class<T> modelClass) {
        if (modelClass.isAssignableFrom(MyViewModel.class)) {
            // Se la classe della ViewModel richiesta è MyViewModel, crea e
            restituisci un'istanza di MyViewModel
            return (T) new MyViewModel();
        }
        throw new IllegalArgumentException("Unknown ViewModel class");
    }
}
```

2. **Inizializza la `ViewModel` utilizzando la factory:** Passa un'istanza della factory alla classe `ViewModelProvider`.

```
java
Copia codice
// Dentro un'Activity o Fragment

// Inizializza la tua classe ViewModel utilizzando la factory
MyViewModel viewModel = new ViewModelProvider(this, new
MyViewModelFactory()).get(MyViewModel.class);
```

Assicurati di sostituire `MyViewModel` con il nome della tua classe `ViewModel`.

3. **Utilizza la `ViewModel`:** Ora puoi utilizzare l'oggetto `viewModel` come al solito per accedere ai campi dati e ai metodi definiti nella tua classe `ViewModel`.

```
java
Copia codice
// Esempio di utilizzo
viewModel.doSomething();
```

Utilizzando una factory, hai maggiore flessibilità nella creazione della `ViewModel`. Puoi implementare la factory in modo da personalizzare la logica di creazione in base a determinate condizioni, come l'utilizzo di parametri personalizzati o l'inizializzazione con dipendenze esterne.

UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: come posso trasformare una risposta api json in gson

RISPOSTA: Per trasformare una risposta API JSON in oggetti Gson in Java, puoi seguire questi passaggi:

1. **Recupera la risposta API:** Utilizza un client HTTP per effettuare una richiesta all'API e ottenere la risposta in formato JSON.
2. **Definisci la struttura dei dati:** Crea classi Java che riflettano la struttura dei dati JSON che ricevi. Queste classi devono avere gli stessi nomi dei campi JSON e i tipi di dati corrispondenti.
3. **Utilizza Gson per il parsing:** Gson è una libreria Java per la serializzazione/deserializzazione di oggetti Java in/da JSON. Utilizzando Gson, puoi convertire la risposta JSON in oggetti Java.

Ecco un esempio di come puoi farlo:

Supponiamo che la risposta API sia questa:

```
json
Copia codice
{
  "nome": "Mario",
  "cognome": "Rossi",
  "età": 30
}
```

Puoi creare una classe Java per rappresentare questi dati:

```
java
Copia codice
public class Persona {
    private String nome;
    private String cognome;
    private int età;

    // Aggiungi i costruttori, i getter e i setter
}
```

Successivamente, utilizzando Gson, puoi convertire la risposta JSON in un oggetto `Persona`:

```
java
Copia codice
import com.google.gson.Gson;

public class Main {
    public static void main(String[] args) {
        String json = "{\"nome\":\"Mario\",\"cognome\":\"Rossi\",\"età\":30}";

        Gson gson = new Gson();
        Persona persona = gson.fromJson(json, Persona.class);

        // Ora puoi accedere ai dati dell'oggetto persona
        System.out.println("Nome: " + persona.getNome());
        System.out.println("Cognome: " + persona.getCognome());
    }
}
```

```
        System.out.println("Età: " + persona.getEtà());
    }
}
```

Assicurati di includere la dipendenza per Gson nel tuo progetto. Se stai utilizzando Maven, puoi aggiungere la dipendenza di Gson nel tuo file `pom.xml`:

```
xml
Copia codice
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.8</version>
</dependency>
```

Se stai utilizzando Gradle, puoi aggiungere la dipendenza di Gson nel tuo file `build.gradle`:

```
groovy
Copia codice
dependencies {
    implementation 'com.google.code.gson:gson:2.8.8'
}
```

Assicurati di sostituire `2.8.8` con l'ultima versione disponibile di Gson al momento dell'implementazione.

RISPOSTA PARZIALMENTE UTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: e utilizzando chiamate retrofit

RISPOSTA: Se stai utilizzando Retrofit per effettuare chiamate API in Android, puoi combinare Retrofit con Gson per gestire automaticamente la conversione da JSON agli oggetti Java.

Ecco un esempio di come farlo:

1. **Definisci un modello di dati:** Come prima cosa, devi definire una classe Java che rappresenta la struttura dei dati JSON che ricevi dall'API. Ad esempio, se stai ricevendo dati su una persona, potrebbe essere una classe `Persona`.
2. **Configura Retrofit:** Configura Retrofit per effettuare la chiamata all'API e ricevere la risposta JSON.
3. **Aggiungi GsonConverterFactory:** Aggiungi `GsonConverterFactory` a Retrofit per gestire la conversione da JSON agli oggetti Java utilizzando Gson.

Ecco un esempio di codice che illustra questo processo:

```
java
Copia codice
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

public class RetrofitClient {
    private static Retrofit retrofit;
    private static final String BASE_URL = "https://api.example.com/";

    public static Retrofit getRetrofitInstance() {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}
```

Successivamente, definisci un'interfaccia Retrofit che descrive le chiamate API che vuoi effettuare:

```
java
Copia codice
import retrofit2.Call;
import retrofit2.http.GET;

public interface ApiService {
    @GET("endpoint")
    Call<Persona> getPersona();
}
```

Ora puoi utilizzare Retrofit per effettuare la chiamata API e ottenere direttamente un oggetto `Persona`:

```
java
Copia codice
```



```

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;

public class Main {
    public static void main(String[] args) {
        ApiService service =
RetrofitClient.getRetrofitInstance().create(ApiService.class);

        Call<Persona> call = service.getPersona();
        call.enqueue(new Callback<Persona>() {
            @Override
            public void onResponse(Call<Persona> call, Response<Persona>
response) {
                if (response.isSuccessful()) {
                    Persona persona = response.body();
                    // Ora puoi accedere ai dati dell'oggetto persona
                    System.out.println("Nome: " + persona.getNome());
                    System.out.println("Cognome: " + persona.getCognome());
                    System.out.println("Età: " + persona.getEtà());
                } else {
                    System.out.println("Errore nella risposta API");
                }
            }

            @Override
            public void onFailure(Call<Persona> call, Throwable t) {
                System.out.println("Errore nella chiamata API: " +
t.getMessage());
            }
        });
    }
}

```

Assicurati di sostituire `https://api.example.com/` con l'URL effettivo del tuo endpoint API, e endpoint con il percorso specifico dell'endpoint che vuoi chiamare.

Con questo setup, Retrofit si occupa della chiamata API e Gson si occupa della conversione della risposta JSON in un oggetto `Persona`.

PARZIALMENTE UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: e come faccio a salvare i dati gson nelle giuste variabili java?

RISPOSTA: Per salvare i dati Gson nelle giuste variabili Java, devi assicurarti che le classi Java corrispondano alla struttura dei dati JSON che ricevi dall'API. Puoi fare ciò definendo classi Java con campi corrispondenti ai campi JSON e utilizzando le annotazioni Gson per mapparle correttamente.

Ad esempio, se ricevi un JSON come questo:

```
json
Copia codice
{
  "nome": "Mario",
  "cognome": "Rossi",
  "età": 30
}
```

Puoi definire una classe Java `Persona` corrispondente:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;

public class Persona {
    @SerializedName("nome")
    private String nome;

    @SerializedName("cognome")
    private String cognome;

    @SerializedName("età")
    private int età;

    // Aggiungi i costruttori, i getter e i setter
}
```

Le annotazioni `@SerializedName` sono utilizzate per mappare i nomi dei campi JSON ai nomi dei campi Java.

Quando ricevi la risposta JSON e la deserializzi utilizzando Gson, i dati verranno automaticamente salvati nelle variabili Java corrispondenti. Ad esempio:

```
java
Copia codice
public class Main {
    public static void main(String[] args) {
        String json = "{\"nome\":\"Mario\",\"cognome\":\"Rossi\",\"età\":30}";

        Gson gson = new Gson();
        Persona persona = gson.fromJson(json, Persona.class);

        // Ora puoi accedere ai dati dell'oggetto persona e usarli come desideri
        System.out.println("Nome: " + persona.getNome());
        System.out.println("Cognome: " + persona.getCognome());
        System.out.println("Età: " + persona.getEtà());
    }
}
```

```
    }  
}
```

In questo esempio, i dati JSON vengono deserializzati in un oggetto `Persona` e i valori vengono assegnati automaticamente alle variabili Java corrispondenti (`nome`, `cognome`, `età`). Puoi quindi utilizzare questi valori come desideri nel tuo codice.

UTILE

RIELABORAZIONE MRGINALE

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: e se mi venisse restituita anche una lista che fa parte di un oggetto?

RISPOSTA: Se la risposta JSON include una lista di oggetti come parte dell'oggetto principale, puoi gestirlo definendo una classe Java corrispondente per la lista e includerla nella classe dell'oggetto principale.

Ad esempio, supponiamo che il tuo JSON abbia questa struttura:

```
json
Copia codice
{
  "nome": "Mario",
  "cognome": "Rossi",
  "età": 30,
  "amici": [
    {"nome": "Luigi", "età": 32},
    {"nome": "Maria", "età": 28}
  ]
}
```

Puoi definire una classe `Persona` che include una lista di `Amico`:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;
import java.util.List;

public class Persona {
    @SerializedName("nome")
    private String nome;

    @SerializedName("cognome")
    private String cognome;

    @SerializedName("età")
    private int età;

    @SerializedName("amici")
    private List<Amico> amici;

    // Aggiungi i costruttori, i getter e i setter
}
```

Definisci anche una classe per l'oggetto `Amico`:

```
java
Copia codice
public class Amico {
    @SerializedName("nome")
    private String nome;

    @SerializedName("età")
    private int età;

    // Aggiungi i costruttori, i getter e i setter
}
```

Quando deserializzi la risposta JSON utilizzando Gson, la lista di amici verrà automaticamente convertita in una lista di oggetti `Amico` e assegnata alla variabile `amici` dell'oggetto `Persona`.

Ecco un esempio di come puoi utilizzare questa struttura:

```
java
Copia codice
public class Main {
    public static void main(String[] args) {
        String json =
            "{\"nome\":\"Mario\",\"cognome\":\"Rossi\",\"età\":30,\"amici\": [{\"nome\":\"Lui gi\",\"età\":32},{\"nome\":\"Maria\",\"età\":28}]}";

        Gson gson = new Gson();
        Persona persona = gson.fromJson(json, Persona.class);

        // Ora puoi accedere ai dati dell'oggetto persona e della lista di amici
        System.out.println("Nome: " + persona.getNome());
        System.out.println("Cognome: " + persona.getCognome());
        System.out.println("Età: " + persona.getEtà());

        List<Amico> amici = persona.getAmici();
        System.out.println("Amici:");
        for (Amico amico : amici) {
            System.out.println("Nome: " + amico.getNome() + ", Età: " +
amico.getEtà());
        }
    }
}
```

In questo esempio, la lista di amici viene deserializzata insieme al resto dei dati dell'oggetto `Persona`, e puoi accedere ad essa tramite il metodo `getAmici()`.

PARZIALMENTE UTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: posso salvare invece solo alcune variabili che mi vengono restituite ignorandone altre?

RISPOSTA: Sì, è possibile ignorare alcune variabili che vengono restituite nel JSON e salvare solo quelle di interesse. Puoi farlo utilizzando l'annotazione `@SerializedName` insieme a Gson per mappare solo i campi desiderati e ignorare gli altri.

Ecco come puoi farlo:

Supponiamo che tu voglia ignorare la variabile `cognome` nel JSON di esempio precedente:

```
json
Copia codice
{
  "nome": "Mario",
  "cognome": "Rossi",
  "età": 30
}
```

Puoi definire la classe `Persona` in questo modo:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;

public class Persona {
    @SerializedName("nome")
    private String nome;

    @SerializedName("età")
    private int età;

    // Aggiungi i costruttori, i getter e i setter
}
```

In questo modo, solo i campi `nome` e `età` verranno mappati nel tuo oggetto `Persona`. Quando utilizzi Gson per deserializzare il JSON, ignorerà il campo `cognome`.

Ecco un esempio di utilizzo:

```
java
Copia codice
public class Main {
    public static void main(String[] args) {
        String json = "{\"nome\":\"Mario\",\"cognome\":\"Rossi\",\"età\":30}";

        Gson gson = new Gson();
        Persona persona = gson.fromJson(json, Persona.class);

        // Ora puoi accedere solo ai dati del nome e dell'età
        System.out.println("Nome: " + persona.getNome());
        System.out.println("Età: " + persona.getEtà());
    }
}
```

In questo esempio, la variabile `cognome` viene ignorata durante la deserializzazione e puoi accedere solo ai dati del nome e dell'età.

UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: e se devo salvare solo un attributo di un oggetto dentro il mio oggetto principale

RISPOSTA: Se devi salvare solo un attributo di un oggetto all'interno del tuo oggetto principale, puoi farlo creando una classe interna per quell'oggetto e includendola come un campo nell'oggetto principale.

Ad esempio, supponiamo che il JSON restituito contenga un oggetto annidato `Indirizzo`, ma tu vuoi salvare solo il campo `città` di quell'oggetto all'interno della tua classe `Persona`.

Ecco come potresti farlo:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;

public class Persona {
    @SerializedName("nome")
    private String nome;

    @SerializedName("età")
    private int età;

    @SerializedName("indirizzo")
    private Indirizzo indirizzo;

    public static class Indirizzo {
        @SerializedName("città")
        private String città;

        // Aggiungi i costruttori, i getter e i setter se necessario
    }

    // Aggiungi i costruttori, i getter e i setter
}
```

In questo esempio, definiamo una classe interna `Indirizzo` che contiene solo il campo `città`. All'interno della classe `Persona`, abbiamo un campo `indirizzo` che è un oggetto di tipo `Indirizzo`.

Quando deserializzi il JSON, Gson ignorerà gli altri campi dell'oggetto `Indirizzo` e salverà solo il campo `città` all'interno dell'oggetto `Persona`.

Ecco un esempio di utilizzo:

```
java
Copia codice
public class Main {
    public static void main(String[] args) {
        String json =
            "{\"nome\":\"Mario\", \"età\":30, \"indirizzo\":{\"città\":\"Roma\"}}";

        Gson gson = new Gson();
        Persona persona = gson.fromJson(json, Persona.class);
    }
}
```



```
        // Ora puoi accedere solo alla città dell'indirizzo
        System.out.println("Città: " + persona.getIndirizzo().getCittà());
    }
}
```

In questo esempio, accediamo al campo `città` dell'oggetto `Indirizzo` all'interno dell'oggetto `Persona`.

PARZIALMENTE UTILE

RIELABORZIONE MEDIA

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: gli attributi della classe java devono essere nello stesso ordine del json anche usando `SerializedName`

RISPOSTA: No, gli attributi della classe Java non devono necessariamente essere nello stesso ordine dei campi JSON, anche quando si utilizzano le annotazioni `@SerializedName`. Gson è in grado di mappare correttamente i campi JSON ai campi della classe Java, indipendentemente dall'ordine.

Puoi definire gli attributi della classe Java nell'ordine che preferisci e utilizzare l'annotazione `@SerializedName` per mapparli ai corrispondenti campi JSON.

Ad esempio, considera il seguente JSON:

```
json
Copia codice
{
  "età": 30,
  "nome": "Mario",
  "cognome": "Rossi"
}
```

Puoi definire la classe Java `Persona` in questo modo:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;

public class Persona {
    @SerializedName("nome")
    private String nome;

    @SerializedName("cognome")
    private String cognome;

    @SerializedName("età")
    private int età;

    // Aggiungi i costruttori, i getter e i setter
}
```

Anche se gli attributi `nome`, `cognome` e `età` sono nell'ordine diverso rispetto al JSON, Gson li mappa correttamente utilizzando le annotazioni `@SerializedName`. Non è richiesto che gli attributi della classe Java siano nello stesso ordine dei campi JSON.

PARZIALMENTE UTILE

RIELABORAZIONE MARGINALE

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: come posso salvare questo json { "code" : "codice", "product" : { "_id":, "nutriments":, "allergens":, "brands":, "nutriscore_grade":, "packaging":, "product_name":, "serving_size":, "ecoscore_grade":, } }

RISPOSTA: Per salvare questo JSON in Java, devi prima definire una classe che rappresenti la struttura dei dati. Utilizzando Gson, puoi poi deserializzare il JSON in un'istanza di questa classe.

Ecco come potresti definire le classi Java corrispondenti al JSON fornito:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;

public class Prodotto {
    private String code;
    private DettagliProdotto product;

    // Aggiungi i costruttori, i getter e i setter
}

public class DettagliProdotto {
    @SerializedName("_id")
    private String id;

    private Nutriments nutriments;
    private String[] allergens;
    private String brands;

    @SerializedName("nutriscore_grade")
    private String nutriscoreGrade;

    private String packaging;

    @SerializedName("product_name")
    private String productName;

    @SerializedName("serving_size")
    private String servingSize;

    @SerializedName("ecoscore_grade")
    private String ecoscoreGrade;

    // Aggiungi i costruttori, i getter e i setter
}

public class Nutriments {
    // Definisci gli attributi per i valori nutrizionali, ad esempio calorie,
    grassi, carboidrati, etc.
    // Aggiungi i costruttori, i getter e i setter
}
```

Ora puoi utilizzare Gson per deserializzare il JSON in un oggetto Prodotto:

```
java
Copia codice
import com.google.gson.Gson;
```

```

public class Main {
    public static void main(String[] args) {
        String json =
"{\"code\": \"codice\", \"product\": {\"_id\": \"ID_prodotto\", \"nutriments\": {}, \"allergens\": [], \"brands\": \"Marche\", \"nutriscore_grade\": \"A\", \"packaging\": \"Confezione\", \"product_name\": \"Nome_prodotto\", \"serving_size\": \"Dimensione_porzioni\", \"ecoscore_grade\": \"B\"}}";

        Gson gson = new Gson();
        Prodotto prodotto = gson.fromJson(json, Prodotto.class);

        // Ora puoi accedere ai dati dell'oggetto prodotto e dei dettagli del
prodotto
        System.out.println("Code: " + prodotto.getCode());
        DettagliProdotto dettagli = prodotto.getProduct();
        System.out.println("ID: " + dettagli.getId());
        System.out.println("Marca: " + dettagli.getBrands());
        // e così via per gli altri campi

        // Se necessario, puoi accedere ai dettagli nutrizionali
        Nutriments nutrimenti = dettagli.getNutriments();
        // Esempio: System.out.println("Calorie: " + nutrimenti.getCalories());
    }
}

```

Assicurati di aggiungere i costruttori, i getter e i setter appropriati nelle tue classi Java. Inoltre, se il campo `nutriments` contiene una struttura dati complessa, puoi definire una classe separata per rappresentare quei dettagli nutrizionali e includerla come un attributo della classe `DettagliProdotto`.

PARZIALMENTE UTILE

RIELABORAZIONE MEDIA

RISPOSTA RIGENERATA UNA VOLTA

IMPLEMENTAZIONE

DOMANDA; puoi mostrarmi come dovrei fare i costruttori?

RISPOSTA: Certamente! Ecco come potresti definire i costruttori per le

classi `Prodotto`, `DettagliProdotto` e `Nutriments`:

```
java
Copia codice
import com.google.gson.annotations.SerializedName;

public class Prodotto {
    private String code;
    private DettagliProdotto product;

    public Prodotto(String code, DettagliProdotto product) {
        this.code = code;
        this.product = product;
    }

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public DettagliProdotto getProduct() {
        return product;
    }

    public void setProduct(DettagliProdotto product) {
        this.product = product;
    }
}

public class DettagliProdotto {
    @SerializedName("_id")
    private String id;

    private Nutriments nutriments;
    private String[] allergens;
    private String brands;

    @SerializedName("nutriscore_grade")
    private String nutriscoreGrade;

    private String packaging;

    @SerializedName("product_name")
    private String productName;

    @SerializedName("serving_size")
    private String servingSize;

    @SerializedName("ecoscore_grade")
    private String ecoscoreGrade;
```

```
    public DettagliProdotto(String id, Nutrimenti nutrimenti, String[]  
allergens, String brands, String nutriscoreGrade, String packaging, String  
productName, String servingSize, String ecoscoreGrade) {  
        this.id = id;  
        this.nutrimenti = nutrimenti;  
        this.allergens = allergens;  
        this.brands = brands;  
        this.nutriscoreGrade = nutriscoreGrade;  
        this.packaging = packaging;  
        this.productName = productName;  
        this.servingSize = servingSize;  
        this.ecoscoreGrade = ecoscoreGrade;  
    }  
  
    public String getId() {  
        return id;  
    }  
  
    public void setId(String id) {  
        this.id = id;  
    }  
  
    public Nutrimenti getNutrimenti() {  
        return nutrimenti;  
    }  
  
    public void setNutrimenti(Nutrimenti nutrimenti) {  
        this.nutrimenti = nutrimenti;  
    }  
  
    public String[] getAllergens() {  
        return allergens;  
    }  
  
    public void setAllergens(String[] allergens) {  
        this.allergens = allergens;  
    }  
  
    public String getBrands() {  
        return brands;  
    }  
  
    public void setBrands(String brands) {  
        this.brands = brands;  
    }  
  
    public String getNutriscoreGrade() {  
        return nutriscoreGrade;  
    }  
  
    public void setNutriscoreGrade(String nutriscoreGrade) {  
        this.nutriscoreGrade = nutriscoreGrade;  
    }  
  
    public String getPackaging() {  
        return packaging;  
    }  
  
    public void setPackaging(String packaging) {  
        this.packaging = packaging;  
    }  
  
    public String getProductName() {
```

```

        return productName;
    }

    public void setProductName(String productName) {
        this.productName = productName;
    }

    public String getServingSize() {
        return servingSize;
    }

    public void setServingSize(String servingSize) {
        this.servingSize = servingSize;
    }

    public String getEcoscoreGrade() {
        return ecoscoreGrade;
    }

    public void setEcoscoreGrade(String ecoscoreGrade) {
        this.ecoscoreGrade = ecoscoreGrade;
    }
}

public class Nutriments {
    // Definisci gli attributi per i valori nutrizionali, ad esempio calorie,
    grassi, carboidrati, etc.
    // Aggiungi i costruttori, i getter e i setter
}

```

Questi sono costruttori che accettano tutti i campi delle rispettive classi come parametri e li assegnano ai corrispondenti attributi. Assicurati di includere i getter e i setter per gli attributi che desideri accedere o modificare all'interno delle tue classi.

RISPOSTA UTILE

RIELABORAZIONE MARGINALE

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: come posso salvare un json che inizia così { "count": 20, "page": 1, "page_count": 20, "page_size": 24, "products": [{ "_id": "8076809573252",

RISPOSTA: Per salvare un file JSON iniziando con quei dati, puoi utilizzare un linguaggio di programmazione come Python. Ecco un esempio di come farlo:

```
python
Copia codice
import json

# Dati da salvare
data = {
    "count": 20,
    "page": 1,
    "page_count": 20,
    "page_size": 24,
    "products": [
        {
            "_id": "8076809573252",
            # Altri dati del prodotto...
        }
        # Altri prodotti...
    ]
}

# Percorso del file JSON
file_path = "dati.json"

# Salvataggio del file JSON
with open(file_path, "w") as json_file:
    json.dump(data, json_file, indent=4)

print("File JSON salvato con successo!")
```

Assicurati di sostituire # Altri dati del prodotto... con i dettagli dei prodotti che desideri includere e # Altri prodotti... con altri elementi del tuo elenco di prodotti se necessario. Il codice crea un file chiamato "dati.json" nella stessa directory in cui viene eseguito il programma e vi scrive i dati in formato JSON.

RISPOSTA INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: su java

RISPOSTA: In Java, puoi utilizzare la libreria `Gson` per gestire dati JSON. Ecco un esempio di come puoi salvare un file JSON con i dati forniti utilizzando `Gson`:

```
java
Copia codice
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import java.io.FileWriter;
import java.io.IOException;

public class Main {
    public static void main(String[] args) {
        // Dati da salvare
        String json =
            "{\"count\":20,\"page\":1,\"page_count\":20,\"page_size\":24,\"products\": [{\"_id\": \"8076809573252\"}]}";

        // Creazione di un oggetto Gson
        Gson gson = new GsonBuilder().setPrettyPrinting().create();

        try {
            // Percorso del file JSON
            String filePath = "dati.json";

            // Creazione di un FileWriter per scrivere nel file
            FileWriter writer = new FileWriter(filePath);

            // Conversione degli oggetti Java in formato JSON e scrittura nel
file
            gson.toJson(json, writer);

            // Chiusura del writer
            writer.close();

            System.out.println("File JSON salvato con successo!");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

In questo esempio, viene utilizzato un oggetto `Gson` per convertire il JSON in un formato leggibile e quindi scriverlo su un file chiamato "dati.json". Assicurati di aggiungere la libreria `Gson` al tuo progetto Java per utilizzarla correttamente.

PRZIALMENTE UTILE

RIELABORAZIONE MEDIA

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: utilizzando retrofit

RISPOSTA: Se desideri utilizzare Retrofit per inviare richieste HTTP e ricevere dati JSON da un endpoint API, puoi farlo nel seguente modo. Tuttavia, Retrofit non è specificamente utilizzato per salvare dati JSON su file, ma puoi ricevere i dati JSON dall'API e poi utilizzare Gson per salvarli su file come mostrato in precedenza. Ecco un esempio di come potresti farlo:

```
java
Copia codice
import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

import java.io.FileWriter;
import java.io.IOException;

public class Main {
    private static final String BASE_URL = "https://api.example.com/";

    public static void main(String[] args) {
        Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();

        ApiService apiService = retrofit.create(ApiService.class);
        Call<DataResponse> call = apiService.getData();

        call.enqueue(new Callback<DataResponse>() {
            @Override
            public void onResponse(Call<DataResponse> call,
Response<DataResponse> response) {
                if (response.isSuccessful()) {
                    DataResponse dataResponse = response.body();
                    if (dataResponse != null) {
                        saveJsonToFile(dataResponse);
                    } else {
                        System.out.println("Nessun dato ricevuto dalla
risposta");
                    }
                } else {
                    System.out.println("Errore nella risposta: " +
response.code());
                }
            }

            @Override
            public void onFailure(Call<DataResponse> call, Throwable t) {
                System.out.println("Errore nella chiamata: " + t.getMessage());
            }
        });
    }

    private static void saveJsonToFile(DataResponse dataResponse) {
        try {
            Gson gson = new GsonBuilder().setPrettyPrinting().create();
```

```
String json = gson.toJson(dataResponse);

FileWriter writer = new FileWriter("dati.json");
writer.write(json);
writer.close();

System.out.println("File JSON salvato con successo!");
} catch (IOException e) {
    e.printStackTrace();
}
}
```

In questo esempio, definiamo un'interfaccia `ApiService` che specifica il metodo per ottenere i dati dall'API. Retrofit gestisce il lavoro di eseguire la richiesta HTTP e di deserializzare la risposta JSON in un oggetto Java di tipo `DataResponse`. Successivamente, utilizziamo Gson per convertire l'oggetto `DataResponse` in una stringa JSON e salvarlo su un file chiamato "dati.json". Assicurati di adattare `ApiService` e `DataResponse` alle tue esigenze specifiche

RISPOSTA UTILE

RIELABORAZIONE MARGINALE

RISPOSTA NON RIGENERATA

IMPLEMENTAZIONE

DOMANDA: e come dovrebbe essere la classe java che incapsula l'oggetto che ti ho descritto sopra

RISPOSTA: Se stai cercando di creare una classe Java che rappresenta l'oggetto JSON che hai descritto, potrebbe essere qualcosa del genere:

```
java
Copia codice
import java.util.List;

public class MyData {
    private int count;
    private int page;
    private int pageCount;
    private int pageSize;
    private List<Product> products;

    // Costruttore, getter e setter
    public MyData() {}

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public int getPage() {
        return page;
    }

    public void setPage(int page) {
        this.page = page;
    }

    public int getPageCount() {
        return pageCount;
    }

    public void setPageCount(int pageCount) {
        this.pageCount = pageCount;
    }

    public int getPageSize() {
        return pageSize;
    }

    public void setPageSize(int pageSize) {
        this.pageSize = pageSize;
    }

    public List<Product> getProducts() {
        return products;
    }

    public void setProducts(List<Product> products) {
        this.products = products;
    }
}
```

```

    }

    @Override
    public String toString() {
        return "MyData{" +
            "count=" + count +
            ", page=" + page +
            ", pageCount=" + pageCount +
            ", pageSize=" + pageSize +
            ", products=" + products +
            '}';
    }
}

```

E poi definiamo una classe per rappresentare i prodotti:

```

java
Copia codice
public class Product {
    private String _id;

    // Costruttore, getter e setter
    public Product() {}

    public String get_id() {
        return _id;
    }

    public void set_id(String _id) {
        this._id = _id;
    }

    @Override
    public String toString() {
        return "Product{" +
            "_id='" + _id + '\'' +
            '}';
    }
}

```

Assicurati di adattare i campi e i metodi di queste classi alle esigenze specifiche del tuo JSON. Inoltre, potresti dover aggiungere ulteriori campi e metodi in base agli altri dati presenti nel tuo JSON.

RISPOSTA UTILE

RIELABORAZIONE MARGINALE

RISPOSTA NON RIGENERATA

FIXING

DOMANDA: puoi riordinare questo json

```
{
  "code": "8013971006317",
  "product": {
    "_id": "8013971006317",
    "_keywords": [
      "flour",
      "food",
      "pizza",
      "common",
      "cereal",
      "tarwebloem",
      "la",
      "per",
      "zachte",
      "farina",
      "potatoe",
      "focaccia",
      "beverage",
      "plant-based",
      "wheat",
      "and",
      "grassi",
      "molino"
    ],
    "added_countries_tags": [],
    "additives_n": 0,
    "additives_old_n": 0,
    "additives_old_tags": [],
    "additives_original_tags": [],
    "additives_tags": [],
    "allergens": "",
    "allergens_from_ingredients": "",
    "allergens_from_user": "(en)",
    "allergens_hierarchy": [],
    "allergens_lc": "en",
    "allergens_tags": [],
    "amino_acids_tags": [],
    "brands": "Molino Grassi",
    "brands_tags": ["molino-grassi"],
    "categories": "Cibi e bevande a base vegetale, Cibi a base vegetale, Cereali e patate, Cereali e i loro prodotti, Farine, Farine di cereali, Farine di grano, Farine di grano tenero, nl:Zachte tarwebloem",
    "categories_hierarchy": [
      "en:plant-based-foods-and-beverages",
      "en:plant-based-foods",
      "en:cereals-and-potatoes",
      "en:cereals-and-their-products",
      "en:flours",
      "en:cereal-flours",
      "en:wheat-flours",
      "en:common-wheat-flours",
      "nl:Zachte tarwebloem"
    ],
    "categories_lc": "it",
    "categories_old": "Cibi e bevande a base vegetale, Cibi a base vegetale, Cereali e patate, Cereali e i loro prodotti, Farine, Farine di cereali, Farine di grano, Farine di grano tenero, nl:Zachte tarwebloem",
    "categories_properties": {
      "agribalyse_proxy_food_code": "en:9410"
    },
    "categories_properties_tags": [
      "all-products",
      "categories-known",
      "agribalyse-food-code-unknown",
      "agribalyse-proxy-food-code-9410",
      "agribalyse-proxy-food-code-known",
      "ciqual-food-code-unknown",
      "agribalyse-known",
      "agribalyse-9410"
    ],
    "categories_tags": [
      "en:plant-based-foods-and-beverages",
      "en:plant-based-foods",
      "en:cereals-and-potatoes",
      "en:cereals-and-their-products",
      "en:flours",
      "en:cereal-flours",
      "en:wheat-flours",
      "en:common-wheat-flours",
      "nl:zachte-tarwebloem"
    ],
    "checkers_tags": [],
    "cities_tags": [],
    "code": "8013971006317",
    "codes_tags": [
      "code-13",
      "8013971006xxx",
      "801397100xxxx",
      "80139710xxxxx",
      "8013971xxxxxxx",
      "801397xxxxxxx",
      "80139xxxxxxx",
      "8013xxxxxxx",
      "801xxxxxxx",
      "80xxxxxxx",
      "8xxxxxxx"
    ],
    "compared_to_category": "en:common-wheat-flours",
    "complete": 0,
    "completeness": 0.7875,
    "correctors_tags": [
      "blinkin89",
      "openfoodfacts-contributors",
      "telperion87",
      "packbot",
      "yuka.sY2b0xO6T85zoF3NwEKvImFKQ8jRgW7-CDPguBaE3-miKK20c9tV3K6rPag",
      "chevalstar"
    ],
    "countries": "Italy,Netherlands",
    "countries_hierarchy": [
      "en:italy",
      "en:netherlands"
    ],
    "countries_lc": "en",
    "countries_tags": [
      "en:italy",
      "en:netherlands"
    ],
    "created_t": 1606596097,
    "creator": "foodless",
    "data_quality_bugs_tags": [],
    "data_quality_errors_tags": [],
    "data_quality_info_tags": [
      "en:no-packaging-data",
      "en:ingredients-percent-analysis-ok",
      "en:all-but-one-ingredient-with-specified-percent",
      "en:ecoscore-extended-data-not-computed",
      "en:food-groups-1-known",
      "en:food-groups-2-known",
      "en:food-groups-3-unknown"
    ],
    "data_quality_tags": [
      "en:no-packaging-data",
      "en:ingredients-percent-analysis-ok",
      "en:all-but-one-ingredient-with-specified-percent",
      "en:ecoscore-"
    ]
  }
}
```

extended-data-not-computed","en:food-groups-1-known","en:food-groups-2-known","en:food-groups-3-unknown","en:ingredients-unknown-score-above-0","en:ingredients-100-percent-unknown","en:ecoscore-origins-of-ingredients-origins-are-100-percent-unknown","en:ecoscore-packaging-packaging-data-missing","en:ecoscore-production-system-no-label"],["data_quality_warnings_tags":["en:ingredients-unknown-score-above-0","en:ingredients-100-percent-unknown","en:ecoscore-origins-of-ingredients-origins-are-100-percent-unknown","en:ecoscore-packaging-packaging-data-missing","en:ecoscore-production-system-no-label"],["data_sources":"App - Open Food Facts, Apps, App - Yuka","data_sources_tags":["app-open-food-facts","apps","app-yuka"],["debug_param_sorted_langs":["it","nl"],"ecoscore_data":{"adjustments":{"origins_of_ingredients":{"aggregated_origins":[{"epi_score":"0","origin":"en:unknown","percent":100,"transportation_score":0}],"epi_score":0,"epi_value":-5,"origins_from_categories":["en:unknown"],"origins_from_origins_field":["en:unknown"],"transportation_score":0,"transportation_scores":{"ad":0,"al":0,"at":0,"ax":0,"ba":0,"be":0,"bg":0,"ch":0,"cy":0,"cz":0,"de":0,"dk":0,"dz":0,"ee":0,"eg":0,"es":0,"fi":0,"fo":0,"fr":0,"gg":0,"gi":0,"gr":0,"hr":0,"hu":0,"ie":0,"il":0,"im":0,"is":0,"it":0,"je":0,"lb":0,"li":0,"lt":0,"lu":0,"lv":0,"ly":0,"ma":0,"mc":0,"md":0,"me":0,"mk":0,"mt":0,"nl":0,"no":0,"pl":0,"ps":0,"pt":0,"ro":0,"rs":0,"se":0,"si":0,"sj":0,"sk":0,"sm":0,"sy":0,"tn":0,"tr":0,"ua":0,"uk":0,"us":0,"va":0,"world":0,"xk":0},"transportation_value":0,"transportation_values":{"ad":0,"al":0,"at":0,"ax":0,"ba":0,"be":0,"bg":0,"ch":0,"cy":0,"cz":0,"de":0,"dk":0,"dz":0,"ee":0,"eg":0,"es":0,"fi":0,"fo":0,"fr":0,"gg":0,"gi":0,"gr":0,"hr":0,"hu":0,"ie":0,"il":0,"im":0,"is":0,"it":0,"je":0,"lb":0,"li":0,"lt":0,"lu":0,"lv":0,"ly":0,"ma":0,"mc":0,"md":0,"me":0,"mk":0,"mt":0,"nl":0,"no":0,"pl":0,"ps":0,"pt":0,"ro":0,"rs":0,"se":0,"si":0,"sj":0,"sk":0,"sm":0,"sy":0,"tn":0,"tr":0,"ua":0,"uk":0,"us":0,"va":0,"world":0,"xk":0},"value":-5,"values":{"ad":-5,"al":-5,"at":-5,"ax":-5,"ba":-5,"be":-5,"bg":-5,"ch":-5,"cy":-5,"cz":-5,"de":-5,"dk":-5,"dz":-5,"ee":-5,"eg":-5,"es":-5,"fi":-5,"fo":-5,"fr":-5,"gg":-5,"gi":-5,"gr":-5,"hr":-5,"hu":-5,"ie":-5,"il":-5,"im":-5,"is":-5,"it":-5,"je":-5,"lb":-5,"li":-5,"lt":-5,"lu":-5,"lv":-5,"ly":-5,"ma":-5,"mc":-5,"md":-5,"me":-5,"mk":-5,"mt":-5,"nl":-5,"no":-5,"pl":-5,"ps":-5,"pt":-5,"ro":-5,"rs":-5,"se":-5,"si":-5,"sj":-5,"sk":-5,"sm":-5,"sy":-5,"tn":-5,"tr":-5,"ua":-5,"uk":-5,"us":-5,"va":-5,"world":-5,"xk":-5},"warning":"origins_are_100_percent_unknown"},"packaging":{"non_recyclable_and_non_biodegradable_materials":1,"value":-15,"warning":"packaging_data_missing"},"production_system":{"labels":[],"value":0,"warning":"no_label"},"threatened_species":{"}},"agribalyse":{"agribalyse_food_code":"9410","agribalyse_proxy_food_code":"9410","co2_agriculture":0.550938,"co2_consumption":0,"co2_distribution":0.02681524,"co2_packaging":0.0062820739,"co2_processing":0.035941984,"co2_total":0.7195366119,"co2_transportation":0.099559314,"code":"9410","dqr":"2.57","ef_agriculture":0.077994226,"ef_consumption":0,"ef_distribution":0.0076481636,"ef_packaging":0.0010951549,"ef_processing":0.0048177734,"ef_total":0.0994748298,"ef_transportation":0.0079195119,"is_beverage":0,"name_en":"Wheat flour, type 110","name_fr":"Farine de blé tendre ou froment T110","score":97,"version":"3.1"},"grade":"b","grades":{"ad":"b","al":"b","at":"b","ax":"b","ba":"b","be":"b","bg":"b","ch":"b","cy":"b","cz":"b","de":"b","dk":"b","dz":"b","ee":"b

,"eg":"b","es":"b","fi":"b","fo":"b","fr":"b","gg":"b","gi":"b","gr":"b","hr":"b","hu":"b","ie":"b","il":"b","im":"b","is":"b","it":"b","je":"b","lb":"b","li":"b","lt":"b","lu":"b","lv":"b","ly":"b","ma":"b","mc":"b","md":"b","me":"b","mk":"b","mt":"b","nl":"b","no":"b","pl":"b","ps":"b","pt":"b","ro":"b","rs":"b","se":"b","si":"b","sj":"b","sk":"b","sm":"b","sy":"b","tn":"b","tr":"b","ua":"b","uk":"b","us":"b","va":"b","world":"b","xk":"b"},"missing":{"labels":1,"origins":1,"packagings":1},"missing_data_warning":1,"missing_key_data":1,"score":77,"scores":{"ad":77,"al":77,"at":77,"ax":77,"ba":77,"be":77,"bg":77,"ch":77,"cy":77,"cz":77,"de":77,"dk":77,"dz":77,"ee":77,"eg":77,"es":77,"fi":77,"fo":77,"fr":77,"gg":77,"gi":77,"gr":77,"hr":77,"hu":77,"ie":77,"il":77,"im":77,"is":77,"it":77,"je":77,"lb":77,"li":77,"lt":77,"lu":77,"lv":77,"ly":77,"ma":77,"mc":77,"md":77,"me":77,"mk":77,"mt":77,"nl":77,"no":77,"pl":77,"ps":77,"pt":77,"ro":77,"rs":77,"se":77,"si":77,"sj":77,"sk":77,"sm":77,"sy":77,"tn":77,"tr":77,"ua":77,"uk":77,"us":77,"va":77,"world":77,"xk":77},"status":"known"},"ecoscore_grade":"b","ecoscore_score":77,"ecoscore_tags":["b"],"editors_tags":["chevalstar","telperion87","kiliweb","blinkin89","foodless","yuka.sY2b0x06T85zoF3NwEKvlmFKQ8jRgW7-CDPguBaE3-miKK20c9tV3K6rPag","packbot","openfoodfacts-contributors"],"emb_codes":"","emb_codes_tags":[],"entry_dates_tags":["2020-11-28","2020-11","2020"],"expiration_date":"","food_groups":{"en:cereals"},"food_groups_tags":{"en:cereals-and-potatoes","en:cereals"},"generic_name":"","generic_name_it":"","generic_name_nl":"","id":"8013971006317","image_front_small_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.200.jpg","image_front_thumb_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.100.jpg","image_front_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.400.jpg","image_ingredients_small_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_it.21.200.jpg","image_ingredients_thumb_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_it.21.100.jpg","image_ingredients_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_it.21.400.jpg","image_nutrition_small_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.22.200.jpg","image_nutrition_thumb_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.22.100.jpg","image_nutrition_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.22.400.jpg","image_small_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.200.jpg","image_thumb_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.100.jpg","image_url":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.400.jpg","images":{"1":{"sizes":{"100":{"h":75,"w":100},"400":{"h":300,"w":400},"full":{"h":3000,"w":4000}},"uploaded_t":1606596109,"uploader":"foodless"},"2":{"sizes":{"100":{"h":32,"w":100},"400":{"h":130,"w":400},"full":{"h":435,"w":1339}},"uploaded_t":1627833254,"uploader":"blinkin89"},"3":{"sizes":{"100":{"h":97,"w":100},"400":{"h":387,"w":400},"full":{"h":1964,"w":2032}},"uploaded_t":1627833312,"uploader":"openfoodfacts-contributors"},"4":{"sizes":{"100":{"h":100,"w":61},"400":{"h":400,"w":244},"full":{"

h":491,"w":299}}, "uploaded_t":1652337815, "uploader": "kiliweb"}, "5": {"sizes": {"100": {"h":100, "w":50}, "400": {"h":400, "w":200}, "full": {"h":1200, "w":600}}, "uploaded_t":1652337817, "uploader": "kiliweb"}, "front_it": {"angle":0, "coordinates_image_size": "full", "geometry": "0x0--1--1", "imgid": "4", "normalize": null, "rev": "16", "sizes": {"100": {"h":100, "w":61}, "200": {"h":200, "w":122}, "400": {"h":400, "w":244}, "full": {"h":491, "w":299}}, "white_magic": null, "x1": "-1", "x2": "-1", "y1": "-1", "y2": "-1"}, "front_nl": {"angle": "90", "coordinates_image_size": "400", "geometry": "2400x3820-260-0", "imgid": "1", "normalize": null, "rev": "4", "sizes": {"100": {"h":100, "w":63}, "200": {"h":200, "w":126}, "400": {"h":400, "w":251}, "full": {"h":3820, "w":2400}}, "white_magic": null, "x1": "26", "x2": "266", "y1": "0", "y2": "382"}, "ingredients_it": {"angle": "0", "coordinates_image_size": "full", "geometry": "258x56-187-368", "imgid": "5", "normalize": "false", "rev": "21", "sizes": {"100": {"h":22, "w":100}, "200": {"h":43, "w":200}, "400": {"h":56, "w":258}, "full": {"h":56, "w":258}}, "white_magic": "false", "x1": "187.99996948242188", "x2": "445.9999694824219", "y1": "368.50000762939453", "y2": "424.00000762939453"}, "ingredients_nl": {"angle": null, "coordinates_image_size": "400", "geometry": "0x0-0-0", "imgid": "2", "normalize": null, "rev": "7", "sizes": {"100": {"h":32, "w":100}, "200": {"h":65, "w":200}, "400": {"h":130, "w":400}, "full": {"h":435, "w":1339}}, "white_magic": null, "x1": null, "x2": null, "y1": null, "y2": null}, "nutrition_it": {"angle": "0", "coordinates_image_size": "full", "geometry": "375x378-144-613", "imgid": "5", "normalize": "false", "rev": "22", "sizes": {"100": {"h":100, "w":99}, "200": {"h":200, "w":198}, "400": {"h":378, "w":375}, "full": {"h":378, "w":375}}, "white_magic": "false", "x1": "144.49996948242188", "x2": "519.4999694824219", "y1": "613.5749816894531", "y2": "991.5749816894531"}, "nutrition_nl": {"angle": 0, "coordinates_image_size": "full", "geometry": "0x0--1--1", "imgid": "3", "normalize": null, "rev": "10", "sizes": {"100": {"h":97, "w":100}, "200": {"h":193, "w":200}, "400": {"h":387, "w":400}, "full": {"h":1964, "w":2032}}, "white_magic": null, "x1": "-1", "x2": "-1", "y1": "-1", "y2": "-1"}}, "informers_tags": ["foodless", "blinkin89", "openfoodfacts-contributors", "telperion87", "yuka.sY2b0xO6T85zoF3NwEKvImFKQ8jRgW7-CDPguBaE3-miKK20c9tV3K6rPag", "kiliweb", "chevalstar"], "ingredients": [{"id": "it:harina-de-trigo-blando", "percent_estimate": 100, "percent_max": 100, "percent_min": 100, "text": "Harina de trigo blando"}], "ingredients_analysis": {"en: palm-oil-content-unknown": ["it:harina-de-trigo-blando"], "en: vegan-status-unknown": ["it:harina-de-trigo-blando"], "en: vegetarian-status-unknown": ["it:harina-de-trigo-blando"]}, "ingredients_analysis_tags": ["en: palm-oil-content-unknown", "en: vegan-status-unknown", "en: vegetarian-status-unknown"], "ingredients_from_or_that_may_be_from_palm_oil_n": 0, "ingredients_from_palm_oil_n": 0, "ingredients_from_palm_oil_tags": [], "ingredients_hierarchy": ["it:harina-de-trigo-blando"], "ingredients_n": "1", "ingredients_n_tags": ["1", "1-10"], "ingredients_original_tags": ["it:harina-de-trigo-

blando"],"ingredients_percent_analysis":1,"ingredients_tags":["it:harina-de-trigo-blando"],"ingredients_text":"Harina de trigo blando","ingredients_text_it":"Harina de trigo blando","ingredients_text_nl":"Zachte tarwebloem","ingredients_text_with_allergens":"Harina de trigo blando","ingredients_text_with_allergens_it":"Harina de trigo blando","ingredients_text_with_allergens_nl":"Zachte tarwebloem","ingredients_that_may_be_from_palm_oil_n":0,"ingredients_that_may_be_from_palm_oil_tags":[],"ingredients_with_specified_percent_n":0,"ingredients_with_specified_percent_sum":0,"ingredients_with_unspecified_percent_n":1,"ingredients_with_unspecified_percent_sum":100,"ingredients_without_ciqua_codes":["it:harina-de-trigo-blando"],"ingredients_without_ciqua_codes_n":1,"interface_version_created":"20150316.jqm2","interface_version_modified":"20190830","known_ingredients_n":0,"labels":"","labels_hierarchy":[],"labels_lc":"it","labels_old":"","labels_tags":[],"lang":"it","languages":{"en:dutch":5,"en:italian":5},"languages_codes":{"it":5,"nl":5},"languages_hierarchy":["en:dutch","en:italian"],"languages_tags":["en:dutch","en:italian","en:2","en:multilingual"],"last_edit_dates_tags":["2022-05-12","2022-05","2022"],"last_editor":"chevalstar","last_image_dates_tags":["2022-05-12","2022-05","2022"],"last_image_t":1652337817,"last_modified_by":"chevalstar","last_modified_t":1652344174,"last_updated_t":1707822535,"lc":"it","link":"","main_countries_tags":[],"manufacturing_places":"Molino Grassi - Via Emilia Ovest 347 - 43126 - Parma (PR) - Italia","manufacturing_places_tags":["molino-grassi-via-emilia-ovest-347-43126-parma-pr-italia"],"max_imgid":"5","minerals_tags":[],"misc_tags":["en:nutriscore-computed","en:nutrition-fruits-vegetables-nuts-from-category","en:nutrition-fruits-vegetables-nuts-from-category-en-common-wheat-flours","en:nutrition-fruits-vegetables-legumes-from-category","en:nutrition-fruits-vegetables-legumes-from-category-en-common-wheat-flours","en:nutriscore-2021-same-as-2023","en:nutriscore-2021-a-2023-a","en:packagings-number-of-components-0","en:packagings-not-complete","en:packagings-empty","en:ecoscore-extended-data-not-computed","en:ecoscore-missing-data-warning","en:ecoscore-missing-data-labels","en:ecoscore-missing-data-origins","en:ecoscore-missing-data-packagings","en:ecoscore-missing-data-no-packagings","en:ecoscore-computed"],"no_nutrition_data":"","nova_group_debug":"no nova group if too many ingredients are unknown: 1 out of 1","nova_group_error":"too_many_unknown_ingredients","nova_groups_tags":["unknown"],"nucleotides_tags":[],"nutrient_levels":{"fat":"low","salt":"low","saturated-fat":"low","sugars":"low"},"nutrient_levels_tags":["en:fat-in-low-quantity","en:saturated-fat-in-low-quantity","en:sugars-in-low-quantity","en:salt-in-low-quantity"],"nutriments":{"carbohydrates":69,"carbohydrates_100g":69,"carbohydrates_unit":"g","carbohydrates_value":69,"energy":1418,"energy-kcal":339,"energy-kcal_100g":339,"energy-kcal_unit":"kcal","energy-kcal_value":339,"energy-kcal_value_computed":338.9,"energy_100g":1418,"energy_unit":"kcal","energy_value":339,"fat":1.1,"fat_100g":1.1,"fat_unit":"g","fat_value":1.1,"fiber":2.5,"fiber_100g":2.

5,"fiber_unit":"g","fiber_value":2.5,"fruits-vegetables-legumes-estimate-from-ingredients_100g":0,"fruits-vegetables-legumes-estimate-from-ingredients_serving":0,"fruits-vegetables-nuts-estimate-from-ingredients_100g":0,"fruits-vegetables-nuts-estimate-from-ingredients_serving":0,"nutrition-score-fr":-3,"nutrition-score-fr_100g":-3,"proteins":12,"proteins_100g":12,"proteins_unit":"g","proteins_value":12,"salt":0,"salt_100g":0,"salt_unit":"g","salt_value":0,"saturated-fat":0.2,"saturated-fat_100g":0.2,"saturated-fat_unit":"g","saturated-fat_value":0.2,"sodium":0,"sodium_100g":0,"sodium_unit":"g","sodium_value":0,"sugars":1.3,"sugars_100g":1.3,"sugars_unit":"g","sugars_value":1.3},"nutriscore":{"2021":{"category_available":1,"data":{"energy":1418,"energy_points":4,"energy_value":1418,"fiber":2.5,"fiber_points":2,"fiber_value":2.5,"fruits_vegetables_nuts_colza_walnut_olive_oils":"0","fruits_vegetables_nuts_colza_walnut_olive_oils_points":0,"fruits_vegetables_nuts_colza_walnut_olive_oils_value":0,"is_beverage":0,"is_cheese":0,"is_fat":0,"is_water":0,"negative_points":4,"positive_points":7,"proteins":12,"proteins_points":5,"proteins_value":12,"saturated_fat":0.2,"saturated_fat_points":0,"saturated_fat_value":0.2,"sodium":0,"sodium_points":0,"sodium_value":0,"sugars":1.3,"sugars_points":0,"sugars_value":1.3},"grade":"a","nutrients_available":1,"nutriscore_applicable":1,"nutriscore_computed":1,"score":-3},"2023":{"category_available":1,"data":{"components":{"negative":[{"id":"energy","points":4,"points_max":10,"unit":"kJ","value":1418},{id":"sugars","points":0,"points_max":15,"unit":"g","value":1.3},{id":"saturated_fat","points":0,"points_max":10,"unit":"g","value":0.2},{id":"salt","points":0,"points_max":20,"unit":"g","value":0}],positive":[{"id":"proteins","points":4,"points_max":7,"unit":"g","value":12},{id":"fiber","points":0,"points_max":5,"unit":"g","value":2.5},{id":"fruits_vegetables_legumes","points":0,"points_max":5,"unit":"%","value":0}]},"count_proteins":1,"count_proteins_reason":"negative_points_less_than_11","is_beverage":0,"is_cheese":0,"is_fat_oil_nuts_seeds":0,"is_red_meat_product":0,"is_water":0,"negative_points":4,"negative_points_max":55,"positive_nutrients":["proteins","fiber","fruits_vegetables_legumes"],"positive_points":4,"positive_points_max":17},"grade":"a","nutrients_available":1,"nutriscore_applicable":1,"nutriscore_computed":1,"score":0},"nutriscore_2021_tags":["a"],"nutriscore_2023_tags":["a"],"nutriscore_data":{"energy":1418,"energy_points":4,"energy_value":1418,"fiber":2.5,"fiber_points":2,"fiber_value":2.5,"fruits_vegetables_nuts_colza_walnut_olive_oils":"0","fruits_vegetables_nuts_colza_walnut_olive_oils_points":0,"fruits_vegetables_nuts_colza_walnut_olive_oils_value":0,"grade":"a","is_beverage":0,"is_cheese":0,"is_fat":0,"is_water":0,"negative_points":4,"positive_points":7,"proteins":12,"proteins_points":5,"proteins_value":12,"saturated_fat":0.2,"saturated_fat_points":0,"saturated_fat_value":0.2,"score":-3,"sodium":0,"sodium_points":0,"sodium_value":0,"sugars":1.3,"sugars_points":0,"sugars_value":1.3},"nutriscore_grade":"a","nutriscore_score":-3,"nutriscore_score_opposite":3,"nutriscore_tags":["a"],"nutriscore_version":"2021","nutrition_data":"on","nutrition_data_per":"100g","nutrition_data_prepared":"","nutrition_data_prepared_per":"100g","nutrition_grade_fr":"a","nutrition_grades":"a","nutrition_grades_tags":["a"],"nutrition_score_beverage":0,"nutrition_score_debug":"","obsolete":"","obsolete_since_date":"","origin":"","origin_it":"","origin_nl":"","origins":"","

"origins_hierarchy":[],"origins_lc":"en","origins_tags":[],"other_nutritional_substances_tags":[],"packaging":"Paper","packaging_hierarchy":["en:paper"],"packaging_lc":"en","packaging_materials_tags":[],"packaging_old":"fr:Papier","packaging_old_before_taxonomization":"Papier","packaging_recycling_tags":[],"packaging_shapes_tags":[],"packaging_tags":["en:paper"],"packaging_text":"","packaging_text_it":"","packaging_text_nl":"","packagings":[],"packagings_materials":{},"photographers_tags":["foodless","blinkin89","openfoodfacts-contributors","kiliweb"],"pnns_groups_1":"Cereals and potatoes","pnns_groups_1_tags":["cereals-and-potatoes","known"],"pnns_groups_2":"Cereals","pnns_groups_2_tags":["cereals","known"],"popularity_key":20900000004,"popularity_tags":["top-10000-scans-2021","top-75-percent-scans-2021","top-80-percent-scans-2021","top-85-percent-scans-2021","top-90-percent-scans-2021","top-50-vn-scans-2021","top-100-vn-scans-2021","top-500-vn-scans-2021","top-1000-vn-scans-2021","top-5000-vn-scans-2021","top-10000-vn-scans-2021","top-50000-vn-scans-2021","top-100000-vn-scans-2021","top-country-vn-scans-2021","top-5000-lu-scans-2021","top-10000-lu-scans-2021","top-50000-lu-scans-2021","top-100000-lu-scans-2021"],"product_name":"La farina per pizza e focaccia","product_name_it":"La farina per pizza e focaccia","product_name_nl":"Tipo 00 Pizza & focaccia","product_quantity":"1000","purchase_places":"","purchase_places_tags":[],"quantity":"1000","removed_countries_tags":[],"rev":23,"scans_n":5,"selected_images":{"front":{"display":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.400.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_nl.4.400.jpg"},"small":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.200.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_nl.4.200.jpg"},"thumb":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.16.100.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_nl.4.100.jpg"},"ingredients":{"display":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_it.21.400.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_nl.7.400.jpg"},"small":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_it.21.200.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_nl.7.200.jpg"},"thumb":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_it.21.100.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_nl.7.100.jpg"},"nutrition":{"display":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.22.400.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_nl.10.400.jpg"},"small":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.22.200.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_nl.10.200.jpg"},"thumb":{"it":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.22.100.jpg","nl":"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_nl.10.100.jpg"}}},"sortkey":1606596111,"st

```
ates":"en:to-be-completed, en:nutrition-facts-completed, en:ingredients-completed,
en:expiration-date-to-be-completed, en:packaging-code-to-be-completed,
en:characteristics-to-be-completed, en:origins-to-be-completed, en:categories-
completed, en:brands-completed, en:packaging-completed, en:quantity-completed,
en:product-name-completed, en:photos-to-be-validated, en:packaging-photo-to-be-
selected, en:nutrition-photo-selected, en:ingredients-photo-selected, en:front-
photo-selected, en:photos-uploaded","states_hierarchy":["en:to-be-
completed","en:nutrition-facts-completed","en:ingredients-
completed","en:expiration-date-to-be-completed","en:packaging-code-to-be-
completed","en:characteristics-to-be-completed","en:origins-to-be-
completed","en:categories-completed","en:brands-completed","en:packaging-
completed","en:quantity-completed","en:product-name-completed","en:photos-to-
be-validated","en:packaging-photo-to-be-selected","en:nutrition-photo-
selected","en:ingredients-photo-selected","en:front-photo-selected","en:photos-
uploaded"],"states_tags":["en:to-be-completed","en:nutrition-facts-
completed","en:ingredients-completed","en:expiration-date-to-be-
completed","en:packaging-code-to-be-completed","en:characteristics-to-be-
completed","en:origins-to-be-completed","en:categories-completed","en:brands-
completed","en:packaging-completed","en:quantity-completed","en:product-name-
completed","en:photos-to-be-validated","en:packaging-photo-to-be-
selected","en:nutrition-photo-selected","en:ingredients-photo-selected","en:front-
photo-selected","en:photos-uploaded"],"stores":"","stores_tags":[],"teams":"pain-au-
chocolat","teams_tags":["pain-au-
chocolat"],"traces":"en:soybeans","traces_from_ingredients":"","traces_from_user":"(
en)
en:soybeans","traces_hierarchy":["en:soybeans"],"traces_lc":"en","traces_tags":["en:s
oybeans"],"unique_scans_n":4,"unknown_ingredients_n":"1","unknown_nutrients_ta
gs":[],"update_key":"20240209","vitamins_tags":[],"status":1,"status_verbose":"pro
duct found"}
```

```
RISPOSTA: {
  "code": "8013971006317",
  "product": {
    "_id": "8013971006317",
    "_keywords": [
      "flour",
      "food",
      "pizza",
      "common",
      "cereal",
      "tarwebloem",
      "la",
      "per",
      "zachte",
      "farina",
      "potatoe",
      "focaccia",
      "beverage",
      "plant-based",
      "wheat",
```

```

    "and",
    "grassi",
    "molino"
  ],
  "added_countries_tags": [],
  "additives_n": 0,
  "additives_old_n": 0,
  "additives_old_tags": [],
  "additives_original_tags": [],
  "additives_tags": [],
  "allergens": "",
  "allergens_from_ingredients": "",
  "allergens_from_user": "(en) ",
  "allergens_hierarchy": [],
  "allergens_lc": "en",
  "allergens_tags": [],
  "amino_acids_tags": [],
  "brands": "Molino Grassi",
  "brands_tags": [
    "molino-grassi"
  ],
  "categories": "Cibi e bevande a base vegetale, Cibi a base vegetale, Cereali e patate, Cereali e i loro prodotti, Farine, Farine di cereali, Farine di grano, Farine di grano tenero, nl:Zachte tarwebloem",
  "categories_hierarchy": [
    "en:plant-based-foods-and-beverages",
    "en:plant-based-foods",
    "en:cereals-and-potatoes",
    "en:cereals-and-their-products",
    "en:flours",
    "en:cereal-flours",
    "en:wheat-flours",
    "en:common-wheat-flours",
    "nl:Zachte tarwebloem"
  ],
  "categories_lc": "it",
  "categories_old": "Cibi e bevande a base vegetale, Cibi a base vegetale, Cereali e patate, Cereali e i loro prodotti, Farine, Farine di cereali, Farine di grano, Farine di grano tenero, nl:Zachte tarwebloem",
  "categories_properties": {
    "agribalyse_proxy_food_code:en": "9410"
  },
  "categories_properties_tags": [
    "all-products",
    "categories-known",
    "agribalyse-food-code-unknown",
    "agribalyse-proxy-food-code-9410",
    "agribalyse-proxy-food-code-known",
    "ciqual-food-code-unknown",
    "agribalyse-known",
    "agribalyse-9410"
  ],
  "categories_tags": [
    "en:plant-based-foods-and-beverages",
    "en:plant-based-foods",
    "en:cereals-and-potatoes",
    "en:cereals-and-their-products",
    "en:flours",
    "en:cereal-flours",
    "en:wheat-flours",
    "en:common-wheat-flours",
    "nl:zachte-tarwebloem"
  ],

```

```

"checkers_tags": [],
"cities_tags": [],
"code": "8013971006317",
"codes_tags": [
  "code-13",
  "8013971006xxx",
  "801397100xxxx",
  "80139710xxxxx",
  "8013971xxxxxx",
  "801397xxxxxxx",
  "80139xxxxxxxx",
  "8013xxxxxxxxx",
  "801xxxxxxxxxx",
  "80xxxxxxxxxxx",
  "8xxxxxxxxxxxx"
],
"compared_to_category": "en:common-wheat-flours",
"complete": 0,
"completeness": 0.7875,
"correctors_tags": [
  "blinkin89",
  "openfoodfacts-contributors",
  "telperion87",
  "packbot",
  "yuka.sY2b0xO6T85zoF3NwEKvImFKQ8jRgW7-CDPguBaE3-miKK20c9tV3K6rPag",
  "chevalstar"
],
"countries": "Italy,Netherlands",
"countries_hierarchy": [
  "en:italy",
  "en:netherlands"
],
"countries_lc": "en",
"countries_tags": [
  "en:italy",
  "en:netherlands"
],
"created_t": 1606596097,
"creator": "foodless",
"data_quality_bugs_tags": [],
"data_quality_errors_tags": [],
"data_quality_info_tags": [
  "en:no-packaging-data",
  "en:ingredients-percent-analysis-ok",
  "en:all-but-one-ingredient-with-specified-percent",
  "en:ecoscore-extended-data-not-computed",
  "en:food-groups-1-known",
  "en:food-groups-2-known",
  "en:food-groups-3-unknown"
],
"data_quality_tags": [
  "en:no-packaging-data",
  "en:ingredients-percent-analysis-ok",
  "en:all-but-one-ingredient-with-specified-percent",
  "en:ecoscore-extended-data-not-computed",
  "en:food-groups-1-known",
  "en:food-groups-2-known",
  "en:food-groups-3-unknown",
  "en:ingredients-unknown-score-above-0",
  "en:ingredients-100-percent-unknown",
  "en:ecoscore-origins-of-ingredients-origins-are-100-percent-unknown",
  "en:ecoscore-packaging-data-missing",
  "en:ecoscore-production-system-no-label"
]

```

```

],
"data_quality_warnings_tags": [
  "en:ingredients-unknown-score-above-0",
  "en:ingredients-100-percent-unknown",
  "en:ecoscore-origins-of-ingredients-origins-are-100-percent-unknown",
  "en:ecoscore-packaging-packaging-data-missing",
  "en:ecoscore-production-system-no-label"
],
"data_sources": "App - Open Food Facts, Apps, App - Yuka",
"data_sources_tags": [
  "app-open-food-facts",
  "apps",
  "app-yuka"
],
"debug_param_sorted_langs": [
  "it",
  "nl"
],
"ecoscore_data": {
  "adjustments": {
    "origins_of_ingredients": {
      "aggregated_origins": [
        {
          "epi_score": "0",
          "origin": "en:unknown",
          "percent": 100,
          "transportation_score": 0
        }
      ],
      "epi_score": 0,
      "epi_value": -5,
      "origins_from_categories": [
        "en:unknown"
      ],
      "origins_from_origins_field": [
        "en:unknown"
      ],
      "transportation_score": 0,
      "transportation_scores": {
        "ad": 0,
        "al": 0,
        "at": 0,
        "ax": 0,
        "ba": 0,
        "be": 0,
        "bg": 0,
        "ch": 0,
        "cy": 0,
        "cz": 0,
        "de": 0,
        "dk": 0,
        "dz": 0,
        "ee": 0,
        "eg": 0,
        "es": 0,
        "fi": 0,
        "fo": 0,
        "fr": 0,
        "gg": 0,
        "gi": 0,
        "gr": 0,
        "hr": 0,
        "hu": 0,

```



```
"ie": 0,
"il": 0,
"im": 0,
"is": 0,
"it": 0,
"je": 0,
"lb": 0,
"li": 0,
"lt": 0,
"lu": 0,
"lv": 0,
"ly": 0,
"ma": 0,
"mc": 0,
"md": 0,
"me": 0,
"mk": 0,
"mt": 0,
"nl": 0,
"no": 0,
"pl": 0,
"ps": 0,
"pt": 0,
"ro": 0,
"rs": 0,
"se": 0,
"si": 0,
"sj": 0,
"sk": 0,
"sm": 0,
"sy": 0,
"tn": 0,
"tr": 0,
"ua": 0,
"uk": 0,
"us": 0,
"va": 0,
"world": 0,
"xk": 0
},
"transportation_value": 0,
"transportation_values": {
  "ad": 0,
  "al": 0,
  "at": 0,
  "ax": 0,
  "ba": 0,
  "be": 0,
  "bg": 0,
  "ch": 0,
  "cy": 0,
  "cz": 0,
  "de": 0,
  "dk": 0,
  "dz": 0,
  "ee": 0,
  "eg": 0,
  "es": 0,
  "fi": 0,
  "fo": 0,
  "fr": 0,
  "gg": 0,
  "gi": 0,
```

```

    "gr": 0,
    "hr": 0,
    "hu": 0,
    "ie": 0,
    "il": 0,
    "im": 0,
    "is": 0,
    "it": 0,
    "je": 0,
    "lb": 0,
    "li": 0,
    "lt": 0,
    "lu": 0,
    "lv": 0,
    "ly": 0,
    "ma": 0,
    "mc": 0,
    "md": 0,
    "me": 0,
    "mk": 0,
    "mt": 0,
    "nl": 0,
    "no": 0,
    "pl": 0,
    "ps": 0,
    "pt": 0,
    "ro": 0,
    "rs": 0,
    "se": 0,
    "si": 0,
    "sj": 0,
    "sk": 0,
    "sm": 0,
    "sy": 0,
    "tn": 0,
    "tr": 0,
    "ua": 0,
    "uk": 0,
    "us": 0,
    "va": 0,
    "world": 0,
    "xk": 0
  }
},
"packaging": {
  "non_recyclable_and_non_biodegradable_materials": 0,
  "packagings": [],
  "score": 0,
  "value": 0
},
"production_system": {
  "labels": [],
  "value": 0
}
},
"grade": "unknown",
"grades": {
  "ad": "unknown",
  "al": "unknown",
  "at": "unknown",
  "ax": "unknown",
  "ba": "unknown",
  "be": "unknown",

```

```
"bg": "unknown",
"ch": "unknown",
"cy": "unknown",
"cz": "unknown",
"de": "unknown",
"dk": "unknown",
"dz": "unknown",
"ee": "unknown",
"eg": "unknown",
"es": "unknown",
"fi": "unknown",
"fo": "unknown",
"fr": "unknown",
"gg": "unknown",
"gi": "unknown",
"gr": "unknown",
"hr": "unknown",
"hu": "unknown",
"ie": "unknown",
"il": "unknown",
"im": "unknown",
"is": "unknown",
"it": "unknown",
"je": "unknown",
"lb": "unknown",
"li": "unknown",
"lt": "unknown",
"lu": "unknown",
"lv": "unknown",
"ly": "unknown",
"ma": "unknown",
"mc": "unknown",
"md": "unknown",
"me": "unknown",
"mk": "unknown",
"mt": "unknown",
"nl": "unknown",
"no": "unknown",
"pl": "unknown",
"ps": "unknown",
"pt": "unknown",
"ro": "unknown",
"rs": "unknown",
"se": "unknown",
"si": "unknown",
"sj": "unknown",
"sk": "unknown",
"sm": "unknown",
"sy": "unknown",
"tn": "unknown",
"tr": "unknown",
"ua": "unknown",
"uk": "unknown",
"us": "unknown",
"va": "unknown",
"world": "unknown",
"xk": "unknown"
},
"status": "unknown",
"warnings": {
  "ad": "missing_critical_data",
  "al": "missing_critical_data",
  "at": "missing_critical_data",
```

```

"ax": "missing_critical_data",
"ba": "missing_critical_data",
"be": "missing_critical_data",
"bg": "missing_critical_data",
"ch": "missing_critical_data",
"cy": "missing_critical_data",
"cz": "missing_critical_data",
"de": "missing_critical_data",
"dk": "missing_critical_data",
"dz": "missing_critical_data",
"ee": "missing_critical_data",
"eg": "missing_critical_data",
"es": "missing_critical_data",
"fi": "missing_critical_data",
"fo": "missing_critical_data",
"fr": "missing_critical_data",
"gg": "missing_critical_data",
"gi": "missing_critical_data",
"gr": "missing_critical_data",
"hr": "missing_critical_data",
"hu": "missing_critical_data",
"ie": "missing_critical_data",
"il": "missing_critical_data",
"im": "missing_critical_data",
"is": "missing_critical_data",
"it": "missing_critical_data",
"je": "missing_critical_data",
"lb": "missing_critical_data",
"li": "missing_critical_data",
"lt": "missing_critical_data",
"lu": "missing_critical_data",
"lv": "missing_critical_data",
"ly": "missing_critical_data",
"ma": "missing_critical_data",
"mc": "missing_critical_data",
"md": "missing_critical_data",
"me": "missing_critical_data",
"mk": "missing_critical_data",
"mt": "missing_critical_data",
"nl": "missing_critical_data",
"no": "missing_critical_data",
"pl": "missing_critical_data",
"ps": "missing_critical_data",
"pt": "missing_critical_data",
"ro": "missing_critical_data",
"rs": "missing_critical_data",
"se": "missing_critical_data",
"si": "missing_critical_data",
"sj": "missing_critical_data",
"sk": "missing_critical_data",
"sm": "missing_critical_data",
"sy": "missing_critical_data",
"tn": "missing_critical_data",
"tr": "missing_critical_data",
"ua": "missing_critical_data",
"uk": "missing_critical_data",
"us": "missing_critical_data",
"va": "missing_critical_data",
"world": "missing_critical_data",
"xk": "missing_critical_data"
}
},
"editors_tags": [

```

```

    "blinkin89",
    "yuka.sY2b0xO6T85zoF3NwEKvlmFKQ8jRgW7-CDPguBaE3-miKK20c9tV3K6rPag",
    "packbot",
    "openfoodfacts-contributors",
    "chevalstar",
    "telperion87"
  ],
  "emb_codes_tags": [],
  "entry_dates_tags": [
    "2020-11",
    "2020-11-28",
    "2020",
    "2020-12",
    "2020-12-04"
  ],
  "environment_impact_level": "not-applicable",
  "environment_impact_level_tags": [
    "not-applicable"
  ],
  "expiration_date": "",
  "food_groups_tags": [
    "en:cereal-grains-and-their-products"
  ],
  "generic_name": "Farina di grano tenero tipo \"00\"",
  "generic_name_it": "Farina di grano tenero tipo \"00\"",
  "id": "8013971006317",
  "image_front_small_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/front\_it.4.200.jpg,
  "image_front_thumb_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/front\_it.4.200.jpg,
  "image_front_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/front\_it.4.400.jpg,
  "image_ingredients_small_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients\_it.6.200.jpg,
  "image_ingredients_thumb_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients\_it.6.200.jpg,
  "image_ingredients_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients\_it.6.400.jpg,
  "image_nutrition_small_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition\_it.6.200.jpg,
  "image_nutrition_thumb_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition\_it.6.200.jpg,
  "image_nutrition_url":
https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition\_it.6.400.jpg,
  "images": {
    "1": {
      "sizes": {
        "100": {
          "h": 100,
          "w": 47
        },
        "400": {
          "h": 400,
          "w": 189
        }
      }
    }
  }
}

```

```
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582027,
  "uploader": "blinkin89"
},
"2": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582062,
  "uploader": "blinkin89"
},
"3": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582103,
  "uploader": "blinkin89"
},
"4": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582142,
  "uploader": "blinkin89"
},
```

```
"5": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582231,
  "uploader": "blinkin89"
},
"ingredients_it": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582132,
  "uploader": "blinkin89"
},
"nutrition_it": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582118,
  "uploader": "blinkin89"
},
"packaging_it": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    }
  }
}
```

```

    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582027,
  "uploader": "blinkin89"
},
"front_it": {
  "sizes": {
    "100": {
      "h": 100,
      "w": 47
    },
    "400": {
      "h": 400,
      "w": 189
    },
    "full": {
      "h": 4160,
      "w": 1968
    }
  },
  "uploaded_t": 1606582062,
  "uploader": "blinkin89"
}
},
"informers_tags": [
  "blinkin89",
  "yuka.sY2b0xO6T85zoF3NwEKvImFKQ8jRgW7-CDPguBaE3-miKK20c9tV3K6rPag",
  "packbot",
  "openfoodfacts-contributors",
  "chevalstar",
  "telperion87"
],
"ingredients_analysis_tags": [],
"ingredients_tags": [
  "en:wheat",
  "en:cereals-and-their-products"
],
"labels_tags": [],
"lang": "it",
"languages_tags": [
  "it",
  "en",
  "fr",
  "de",
  "es",
  "nl",
  "hr",
  "pt",
  "sv"
],
"last_edit_dates_tags": [
  "2020-11-28",
  "2020",
  "2020-12",
  "2020-12-04"
],
"last_image_dates_tags": [
  "2020-11-28",
  "2020",

```



```

    "2020-12",
    "2020-12-04"
  ],
  "nutrient_levels": {
    "fat": "low",
    "salt": "low",
    "saturated-fat": "low",
    "sugars": "low"
  },
  "nutrient_levels_tags": [
    "en:low-fat",
    "en:low-saturated-fat",
    "en:low-sugars",
    "en:low-salt"
  ],
  "nutriments": {
    "carbohydrates": 74.9,
    "carbohydrates_100g": 74.9,
    "carbohydrates_unit": "g",
    "carbohydrates_value": 74.9,
    "energy": 1544,
    "energy-kcal": 364,
    "energy-kcal_100g": 364,
    "energy-kcal_unit": "kcal",
    "energy-kcal_value": 364,
    "energy_100g": 1544,
    "energy_unit": "kcal",
    "energy_value": 364,
    "fat": 1,
    "fat_100g": 1,
    "fat_unit": "g",
    "fat_value": 1,
    "fiber": 3,
    "fiber_100g": 3,
    "fiber_unit": "g",
    "fiber_value": 3,
    "fruits-vegetables-nuts-estimate-from-ingredients_100g": 0,
    "nova-group": 1,
    "nova-group_100g": 1,
    "nova-group_serving": 1,
    "nutrition-score-fr": -2,
    "nutrition-score-fr_100g": -2,
    "proteins": 11,
    "proteins_100g": 11,
    "proteins_unit": "g",
    "proteins_value": 11,
    "salt": 0,
    "salt_100g": 0,
    "salt_unit": "g",
    "salt_value": 0,
    "saturated-fat": 0,
    "saturated-fat_100g": 0,
    "saturated-fat_unit": "g",
    "saturated-fat_value": 0,
    "sodium": 0,
    "sodium_100g": 0,
    "sodium_unit": "g",
    "sodium_value": 0,
    "sugars": 0,
    "sugars_100g": 0,
    "sugars_unit": "g",
    "sugars_value": 0
  },

```

```

"nutriscore_data": {
  "energy": 1544,
  "energy_points": 4,
  "energy_value": 1544,
  "fiber": 3,
  "fiber_points": 3,
  "fiber_value": 3,
  "fruits_vegetables_nuts_colza_walnut_olive_oils": 0,
  "fruits_vegetables_nuts_colza_walnut_olive_oils_points": 0,
  "fruits_vegetables_nuts_colza_walnut_olive_oils_value": 0,
  "negative_points": 4,
  "positive_points": 6,
  "proteins": 11,
  "proteins_points": 6,
  "proteins_value": 11,
  "saturated_fat": 0,
  "saturated_fat_points": 0,
  "saturated_fat_ratio": 0,
  "saturated_fat_ratio_points": 0,
  "saturated_fat_ratio_value": 0,
  "sodium": 0,
  "sodium_points": 0,
  "sodium_value": 0,
  "score": -2,
  "sugars": 0,
  "sugars_points": 0,
  "sugars_value": 0
},
"origins": "",
"packaging": "",
"packaging_tags": [],
"product_name": "Farina di grano tenero tipo \"00\"",
"quantity": "1 kg",
"selected_images": {
  "front": {
    "display": {
      "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.4.40
0.jpg"
    },
    "small": {
      "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.4.20
0.jpg"
    },
    "thumb": {
      "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/front_it.4.10
0.jpg"
    }
  },
  "ingredients": {
    "display": {
      "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_i
t.6.400.jpg"
    },
    "small": {
      "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_i
t.6.200.jpg"
    },
    "thumb": {

```

```

        "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/ingredients_i
t.6.100.jpg"
    },
    "nutrition": {
        "display": {
            "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.
6.400.jpg"
        },
        "small": {
            "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.
6.200.jpg"
        },
        "thumb": {
            "it":
"https://images.openfoodfacts.org/images/products/801/397/100/6317/nutrition_it.
6.100.jpg"
        }
    },
    "sortkey": 1598138530,
    "states_tags": [
        "en:to-be-completed",
        "en:to-be-checked",
        "en:complete",
        "en:in-progress"
    ],
    "stores_tags": [],
    "traces": "",
    "traces_from_ingredients": "",
    "traces_tags": [],
    "unique_scans_n": 1,
    "unknown_nutrients_tags": [],
    "update_key": "packbot"
},
"product_name": "Grano tenero tipo 00",
"ingredients_text": "grano tenero tipo 00",
"quantity": "1kg",
"brands": "Viaggiator Goloso",
"categories": "Alimenti e bevande a base vegetale, Alimenti a base vegetale,
Cibi a base di piante, Cibi a base di cereali e patate, Farine, Farina di
frumento",
"labels": "",
"countries_tags": [
    "it"
],
"nutriments": {
    "carbohydrates_value": 74.9,
    "carbohydrates_unit": "g",
    "carbohydrates_100g": 74.9,
    "carbohydrates": 74.9,
    "energy_value": 364,
    "energy_unit": "kcal",
    "energy_kcal_value": 364,
    "energy_kcal_unit": "kcal",
    "energy_kcal_100g": 364,
    "energy": 364,
    "energy_100g": 364,
    "fat_value": 1,
    "fat_unit": "g",

```

```

"fat_100g": 1,
"fat": 1,
"fiber_value": 3,
"fiber_unit": "g",
"fiber_100g": 3,
"fiber": 3,
"nutrition-score-fr_100g": -2,
"proteins_value": 11,
"proteins_unit": "g",
"proteins_100g": 11,
"proteins": 11,
"salt_value": 0,
"salt_unit": "g",
"salt_100g": 0,
"salt": 0,
"saturated-fat_value": 0,
"saturated-fat_unit": "g",
"saturated-fat_100g": 0,
"saturated-fat": 0,
"sodium_value": 0,
"sodium_unit": "g",
"sodium_100g": 0,
"sodium": 0,
"sugars_value": 0,
"sugars_unit": "g",
"sugars_100g": 0,
"sugars": 0
},
"nutrient_levels": {
  "fat": "low",
  "salt": "low",
  "saturated-fat": "low",
  "sugars": "low"
},
"nutriscore_data": {
  "score": -2,
  "proteins_value": 11,
  "proteins_points": 6,
  "fiber_value": 3,
  "fiber_points": 3,
  "energy_value": 364,
  "energy

```

The product "Grano tenero tipo 00" from the brand Viaggiator Goloso is a type of wheat flour commonly used in Italian cuisine, particularly for baking and pasta making. Here is a detailed breakdown of its nutritional content and other relevant information:

Product Information:

- **Product Name:** Grano tenero tipo 00
- **Brand:** Viaggiator Goloso
- **Quantity:** 1 kg
- **Categories:** Alimenti e bevande a base vegetale (Plant-based foods and beverages), Alimenti a base vegetale (Plant-based foods), Cibi a base di piante (Plant-based foods), Cibi a base di cereali e patate (Cereal and potato-based foods), Farine (Flours), Farina di frumento (Wheat flour)
- **Country of Origin:** Italy

Ingredients:

- **Primary Ingredient:** Grano tenero tipo 00 (Soft wheat type 00)

Nutritional Information (per 100g):

- **Energy:** 364 kcal (1544 kJ)
- **Carbohydrates:** 74.9 g
- **Fat:** 1 g
- **Fiber:** 3 g
- **Proteins:** 11 g
- **Salt:** 0 g
- **Sugars:** 0 g
- **Saturated Fat:** 0 g
- **Sodium:** 0 g

Nutritional Quality:

- **Nutrient Levels:**
 - Fat: Low
 - Saturated Fat: Low
 - Sugars: Low
 - Salt: Low
- **Nutri-Score:** -2 (Indicating a favorable nutritional profile for this type of product)

Environmental and Label Information:

- **Environment Impact Level:** Not applicable
- **Labels and Certifications:** None listed

Images:

- **Front View:**
- **Ingredients:**
- **Nutrition Information:**

This flour is suitable for those looking for a high-carbohydrate, low-fat, and low-sugar ingredient, ideal for baking and cooking needs that require a refined flour with fine texture. It is particularly used in making pasta, pastries, and other delicate baked goods due to its fine milling.

RISPOSTA UTILE

RIELABORAZIONE MARGINALE

RISPOSTA NON RIGENERATA

TESTING

DOMANDA: sto implementando dei test per la mia app android ma ho l'errore cannot resolve symbol espresso cosa posso fare, sto usando java

RISPOSTA: L'errore "Cannot resolve symbol Espresso" indica che il tuo progetto non riesce a trovare la libreria Espresso per i test UI. Per risolvere questo problema, devi assicurarti di aver aggiunto correttamente le dipendenze di Espresso nel tuo file `build.gradle` del modulo dell'app.

Ecco i passaggi che puoi seguire per configurare Espresso nel tuo progetto Android:

1. **Apri il file `build.gradle` del modulo dell'app** (di solito si trova in `app/build.gradle`).
2. **Aggiungi le dipendenze di Espresso** nella sezione `dependencies`. Assicurati che il file `build.gradle` contenga le seguenti dipendenze:

```
gradle
Copia codice
android {
    ...
    testOptions {
        animationsDisabled = true
    }
}

dependencies {
    // Dependency for AndroidX Test
    androidTestImplementation 'androidx.test:core:1.4.0'
    androidTestImplementation 'androidx.test:runner:1.4.0'
    androidTestImplementation 'androidx.test:rules:1.4.0'

    // Dependency for Espresso
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    androidTestImplementation 'androidx.test.espresso:espresso-intents:3.4
```

RISPOSTA INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

TESTING

DOMANDA: continua a darmi errore cosa dovrei importare nella classe di test?

RISPOSTA: Se hai aggiunto correttamente le dipendenze nel file `build.gradle` ma continui a riscontrare l'errore, assicurati di importare correttamente le classi di Espresso nella tua classe di test.

Ecco un esempio di come impostare una classe di test con Espresso in Java:

1. **Verifica che le dipendenze nel `build.gradle` siano corrette.** Dovrebbero assomigliare a questo:

```
gradle
Copia codice
android {
    ...
    testOptions {
        animationsDisabled = true
    }
}

dependencies {
    // Dependency for AndroidX Test
    androidTestImplementation 'androidx.test:core:1.4.0'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test:runner:1.4.0'
    androidTestImplementation 'androidx.test:rules:1.4.0'

    // Dependency for Espresso
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    androidTestImplementation 'androidx.test.espresso:espresso-
intents:3.4.0'
    androidTestImplementation 'androidx.test.espresso:espresso-
contrib:3.4.0'
}
```

2. **Importa correttamente le classi nella tua classe di test.** Ecco un esempio di una classe di test:

```
java
Copia codice
package com.example.myapp;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;
import androidx.test.espresso.Espresso;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;

@RunWith(AndroidJUnit4.class)
public class MyActivityTest {
```

```

@Rule
public ActivityTestRule<MyActivity> activityRule =
    new ActivityTestRule<>(MyActivity.class);

@Test
public void testSomeUiElement() {
    // Use Espresso to perform actions and checks
    onView(withId(R.id.my_text_view))
        .check(matches(withText("Hello World!")));
}
}

```

3. **Assicurati di sincronizzare il progetto con Gradle** dopo aver aggiunto o modificato le dipendenze nel file `build.gradle`. Puoi farlo cliccando su "Sync Now" che appare in alto a destra nell'IDE, oppure andando su `File > Sync Project with Gradle Files`.
4. **Pulisci e ricostruisci il progetto.** Vai su `Build > Clean Project` e poi su `Build > Rebuild Project`.
5. **Verifica che gli import siano corretti.** Dovresti avere import simili a questi nella tua classe di test:

```

java
Copia codice
import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;
import androidx.test.espresso.Espresso;
import androidx.test.espresso.assertion.ViewAssertions;
import androidx.test.espresso.matcher.ViewMatchers;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;

```

Se hai seguito tutti questi passaggi e ancora ricevi l'errore, potresti voler invalidare la cache e riavviare Android Studio:

1. Vai su `File > Invalidate Caches / Restart`.
2. Seleziona `Invalidate and Restart`.

Questi passaggi dovrebbero aiutarti a risolvere l'errore "Cannot resolve symbol Espresso" nel tuo progetto Android.

RISPOSTA INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

TESTING

DOMANDA: continua a non funzionare

RISPOSTA: Se continui a riscontrare l'errore "Cannot resolve symbol Espresso", ci sono alcuni ulteriori passaggi che puoi provare per risolvere il problema. Potrebbe essere necessario controllare alcuni dettagli aggiuntivi nel tuo setup.

Passaggi Aggiuntivi per Risolvere l'Errore

1. **Verifica le Versioni delle Dipendenze:** Assicurati che le versioni delle dipendenze siano compatibili tra loro e con la versione di Android Studio che stai utilizzando. Ecco una configurazione aggiornata:

```
gradle
Copia codice
android {
    ...
    testOptions {
        animationsDisabled = true
    }
}

dependencies {
    // Dependency for AndroidX Test
    androidTestImplementation 'androidx.test:core:1.4.0'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test:runner:1.5.2'
    androidTestImplementation 'androidx.test:rules:1.5.0'

    // Dependency for Espresso
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
intents:3.5.1'
    androidTestImplementation 'androidx.test.espresso:espresso-
contrib:3.5.1'
}
```

2. **Controlla il tuo Repositorio Maven:** Assicurati di avere i repository Maven corretti nel file `build.gradle` a livello di progetto (di solito `build.gradle` nella directory principale del progetto):

```
gradle
Copia codice
allprojects {
    repositories {
        google()
        mavenCentral()
    }
}
```

3. **Forza la Sincronizzazione del Progetto:**
 - o Vai su `File > Sync Project with Gradle Files`.
 - o Oppure, apri il terminale integrato di Android Studio e esegui `./gradlew clean build` per pulire e costruire il progetto.

4. **Verifica la Configurazione di Gradle Wrapper:** Assicurati che il wrapper di Gradle sia aggiornato. Nel file `gradle/wrapper/gradle-wrapper.properties`, dovrebbe apparire qualcosa del genere:

```
properties
Copia codice
distributionUrl=https\://services.gradle.org/distributions/gradle-7.4.2-
bin.zip
```

5. **Esempio Completo di Classe di Test:** Verifica che la tua classe di test includa correttamente gli import. Ecco un esempio completo:

```
java
Copia codice
package com.example.myapp;

import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.assertion.ViewAssertions.matches;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
import static androidx.test.espresso.matcher.ViewMatchers.withText;

@RunWith(AndroidJUnit4.class)
public class MyActivityTest {

    @Rule
    public ActivityTestRule<MyActivity> activityRule =
        new ActivityTestRule<>(MyActivity.class);

    @Test
    public void testSomeUiElement() {
        onView(withId(R.id.my_text_view))
            .check(matches(withText("Hello World!")));
    }
}
```

6. **Pulizia della Cache e Riavvio di Android Studio:**

- Vai su File > Invalidate Caches / Restart.
- Seleziona Invalidate and Restart.

Controlla i Permessi di Internet

Se utilizzi Espresso-Intents, potrebbe essere necessario aggiungere il permesso per Internet nel file `AndroidManifest.xml` del tuo modulo principale:

```
xml
Copia codice
<uses-permission android:name="android.permission.INTERNET" />
```

Prova con un Progetto di Test Minimo

Se ancora non funziona, prova a creare un nuovo progetto di test minimale seguendo questi passaggi:

1. **Crea un nuovo progetto in Android Studio.**
2. **Aggiungi le dipendenze di Espresso nel file `build.gradle`.**
3. **Crea una semplice attività e un test Espresso.**

Questo aiuta a determinare se il problema è specifico del tuo progetto attuale o un problema generale con la configurazione del tuo ambiente di sviluppo.

Se hai seguito tutti questi passaggi e continui a riscontrare l'errore, condividi ulteriori dettagli sull'errore specifico e sulla configurazione del tuo progetto, in modo che possiamo approfondire ulteriormente.

RISPOSTA INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA

TESTING

DOMANDA: durante il test espresso il mio database room si svuota cosa posso fare

RISPOSTA: Se durante l'esecuzione dei test espresso il tuo database Room si svuota, potrebbero esserci diverse cause. Di seguito alcune possibili soluzioni per risolvere il problema:

1. Utilizzo di un database in memoria per i test

Durante i test, potrebbe essere utile utilizzare un database in memoria piuttosto che quello persistente. Questo permette di isolare i dati dei test dal database di produzione.

Esempio di configurazione per un database in memoria:

```
java
Copia codice
@Database(entities = {YourEntity.class}, version = 1)
public abstract class AppDatabase extends RoomDatabase {
    // Your DAOs
}

// In your test setup
AppDatabase db = Room.inMemoryDatabaseBuilder(context,
AppDatabase.class).build();
```

2. Ripristinare lo stato del database prima di ogni test

Assicurati che il database venga ripristinato ad uno stato conosciuto prima di ogni test. Questo può essere fatto nel metodo annotato con `@Before` nella tua classe di test.

Esempio di configurazione:

```
java
Copia codice
@Before
public void setUp() {
    db = Room.inMemoryDatabaseBuilder(context, AppDatabase.class).build();
    // Popola il database con dati di test iniziali
    populateDatabase();
}

@After
public void tearDown() {
    db.close();
}
```

3. Utilizzo delle annotazioni di JUnit correttamente

Assicurati di usare correttamente le annotazioni `@Before` e `@After` per preparare e pulire il database prima e dopo ogni test.

```
java
Copia codice
@Before
public void initDb() {
```

```

        db = Room.inMemoryDatabaseBuilder(
            ApplicationProvider.getApplicationContext(),
            AppDatabase.class)
                .allowMainThreadQueries()
                .build();
    }

    @After
    public void closeDb() {
        db.close();
    }

```

4. Assicurarsi che i dati persistano correttamente

Controlla che i dati vengano correttamente inseriti nel database durante i test. Puoi fare questo verificando il contenuto del database prima e dopo ogni operazione di test.

Esempio di test:

```

java
Copia codice
@Test
public void testInsertAndRead() {
    YourEntity entity = new YourEntity();
    entity.setValue("test");
    db.yourDao().insert(entity);

    List<YourEntity> allEntities = db.yourDao().getAll();
    assertEquals(allEntities.get(0).getValue(), "test");
}

```

5. Debug e Log

Aggiungi log per monitorare le operazioni del database durante i test. Questo può aiutare a identificare dove e perché i dati vengono persi.

Esempio di utilizzo di log:

```

java
Copia codice
Log.d("Test", "Inserimento entità nel database");
db.yourDao().insert(entity);

Log.d("Test", "Recupero entità dal database");
List<YourEntity> allEntities = db.yourDao().getAll();
Log.d("Test", "Numero di entità nel database: " + allEntities.size());

```

6. Controlla il ciclo di vita del contesto

Assicurati che il contesto utilizzato per creare il database sia corretto e che non venga ricreato o eliminato durante i test.

Seguendo questi suggerimenti dovresti essere in grado di identificare e risolvere il problema del database Room che si svuota durante i test Espresso.

RISPOSTA PARZIALMENTE UTILE

RIELABORAIONE MEDIA

RISPOSTA NON RIGENERATA

TESTING

DOMANDA: come posso aprire il primo item di una RecyclerView con espresso java

RISPOSTA: Per aprire il primo item di una `RecyclerView` utilizzando Espresso in Java, devi seguire alcuni passaggi. Espresso è una libreria di testing per Android che permette di interagire con le UI delle app in modo programmatico. Ecco come puoi fare:

1. Aggiungi le dipendenze necessarie nel tuo `build.gradle`:

```
groovy
Copia codice
dependencies {
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    androidTestImplementation 'androidx.test.espresso:espresso-contrib:3.4.0' // Per interagire con RecyclerView
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test:rules:1.4.0'
}
```

2. Assicurati di avere i corretti import nei tuoi test:

```
java
Copia codice
import androidx.test.ext.junit.runners.AndroidJUnit4;
import androidx.test.rule.ActivityTestRule;
import androidx.test.espresso.contrib.RecyclerViewActions;

import org.junit.Rule;
import org.junit.Test;
import org.junit.runner.RunWith;

import static androidx.test.espresso.Espresso.onView;
import static androidx.test.espresso.action.ViewActions.click;
import static androidx.test.espresso.matcher.ViewMatchers.withId;
```

3. Scrivi il test per aprire il primo item della RecyclerView:

```
java
Copia codice
@RunWith(AndroidJUnit4.class)
public class RecyclerViewTest {

    @Rule
    public ActivityTestRule<MainActivity> activityRule =
        new ActivityTestRule<>(MainActivity.class);

    @Test
    public void testOpenFirstItem() {
        // Supponendo che la tua RecyclerView abbia l'id recyclerView
        onView(withId(R.id.recyclerView))
            .perform(RecyclerViewActions.actionOnItemAtPosition(0,
click())));
    }
}
```

Spiegazione dei passaggi:

1. **Dependencies:** Aggiungi le dipendenze necessarie per Espresso e RecyclerView nel file `build.gradle` del modulo `app`.
2. **Imports:** Assicurati di importare le classi necessarie per il testing con Espresso, inclusi i runner di JUnit, le regole per le attività e le azioni specifiche per RecyclerView.
3. **ActivityTestRule:** Definisci una regola per l'attività che vuoi testare. In questo esempio, è `MainActivity`.
4. **Test Method:** Crea un metodo di test annotato con `@Test`. Utilizza il metodo `onView(withId(R.id.recyclerView))` per trovare la tua RecyclerView. Poi usa `perform(RecyclerViewActions.actionOnItemAtPosition(0, click()))` per cliccare sul primo elemento della lista (indice 0).

Note:

- Assicurati che l'ID della tua RecyclerView corrisponda a quello definito nel tuo layout (`R.id.recyclerView` in questo esempio).
- Puoi eseguire questo test su un dispositivo fisico o un emulatore configurato per i test strumentali.

Questo è un esempio di base. Puoi espandere i tuoi test per verificare anche altri comportamenti della tua RecyclerView e dei suoi elementi.

RISPOSTA INUTILE

RIELABORAZIONE TOTALE

RISPOSTA NON RIGENERATA