



Documentazione

Progettazione e implementazione di un sistema per la gestione di un e-commerce per una realtà aziendale esistente e attiva nell’ambito di agronomia e servizi di consulenza.

Progetto di WEB COMPUTING

A.A. 2020/2021

Matteo Perfidio – 200632

Antonino Natale – 200705

Davide Crisafulli – 195097

Sommario

[Introduzione 2](#_Toc64728474)

[Descrizione 3](#_Toc64728475)

[API 3](#_Toc64728476)

[Auth 4](#_Toc64728477)

[Services 4](#_Toc64728478)

[Models 4](#_Toc64728479)

[Persistence 5](#_Toc64728480)

[Components 5](#_Toc64728481)

[Web 6](#_Toc64728482)

[Views 6](#_Toc64728483)

[Locale 6](#_Toc64728484)

[Logging 6](#_Toc64728485)

[Schema E-R 7](#_Toc64728486)

[Diagramma UML delle Classi 8](#_Toc64728487)

[Tecnologie Utilizzate 9](#_Toc64728488)

[Back-end 9](#_Toc64728489)

[Front-End 9](#_Toc64728490)

[Indice 10](#_Toc64728491)

[Figure 10](#_Toc64728492)

# Introduzione

L’applicativo web sviluppato è uno strumento pensato per consentire la vendita di servizi e prodotti da parte di una realtà aziendale esistente: *BioAgri S.r.l.s.* - azienda che opera nel settore agrario distribuendo ed erogando servizi di consulenza e prodotti per il giardinaggio, coltura, e strumenti per l’agronomia.

Il progetto prevede uno sviluppo di un’architettura modello **Client-Server**, con standard di comunicazione **RESTful** e supporto multilingua per una fruizione multipiattaforma di un’applicazione web orientata allo **shopping online** e all’ingresso della stessa nel mercato digitale globale.

Tramite un’interfaccia molto semplice e intuitiva l’utente ha la possibilità di visionare una vetrina che evidenzia le principali informazioni riguardo la sopracitata azienda e in seguito accedere al catalogo dei prodotti, per mezzo del quale effettuare eventuali operazioni d’acquisto.

## Descrizione

L’architettura dell’applicativo è divisa in due blocchi architetturalmente indipendenti tra loro: back-end e front-end; messi in comunicazione tramite API. L’intento è quello di creare un applicativo multipiattaforma, RESTful e flessibile all’evoluzione della stessa.



Figura 14

### API

Interfaccia di comunicazione omogenea basata su architettura REST che mette in comunicazione back-end e i relativi front-end garantendo e implementando meccanismi di sicurezza - attraverso permessi e autenticazioni, meccanismi di manipolazione delle risorse e meccanismi di caching per migliorare l’efficienza computazionale.

Fintanto che l’interfaccia non viene modificata client e server possono essere sostituiti e sviluppati indipendentemente l’uno dall’altro, ciò permette di semplificare e disaccoppiare l’architettura in blocchi perfettamente distinti.

### Auth

Protocollo di autenticazione per l’utilizzo delle API con accesso ristretto.  
Il client comunica l’intenzione al server di autenticarsi con username e password forniti dall’Utente ottenendo un **token di accesso** temporaneo da utilizzare come chiave di validazione per le proprie richieste.  
Un **token di accesso** è valido solo per un breve periodo di tempo, quindi il sistema di autenticazione emetterà un token di aggiornamento che il client sostituirà al suo precedente, previa perdita dei privilegi acquisiti.

### Services

Collezione di entità che descrivono e manipolano le interazioni con servizi esterni:

#### Servizi di pagamento

* PayPal

#### Servizi di accesso

* Google
* Facebook
* Twitter

### Models

La collezione di entità che descrivono il modello relazionale della base di dati conformandosi al linguaggio di programmazione utilizzato.

### Persistence

Per la gestione della persistenza l’applicativo si affida ad un DBMS relazionale: PostgreSQL.

### Components

Entità per la gestione e manipolazione dell’intero ciclo di vita degli oggetti grafici.  
I componenti si suddividono in due tipologie:

* *Stateless: senza stato, statici;*
* *Stateful: con stato, dinamici;*

I primi non possiedono uno stato, durante il loro ciclo di vita non vi è alcun tipo di cambiamento.   
I secondi possiedono uno stato manipolabile, durante il loro ciclo di vita possono subire delle modifiche al loro stato intrinseco che riflette i cambiamenti all’esterno.

Ogni componente è dotato di un’interfaccia MVC attraverso la quale viene delegata la sua intera logica applicativa.

Ogni componente inoltre può essere riutilizzato, duplicato e innestato all’interno di altri componenti, con i quali vi è possibile interagire e comunicare.

### Web

Collezione di entità per la gestione delle comunicazioni su web tramite protocollo HTTP.

### Views

Collezione di entità per la visualizzazione grafica su web browser dell’applicativo.

### Locale

Sistema di gestione dalla lingua di visualizzazione.

### Logging

Sistema di tracciamento delle azioni e degli eventi all’interno dell’applicativo.

# Schema E-R

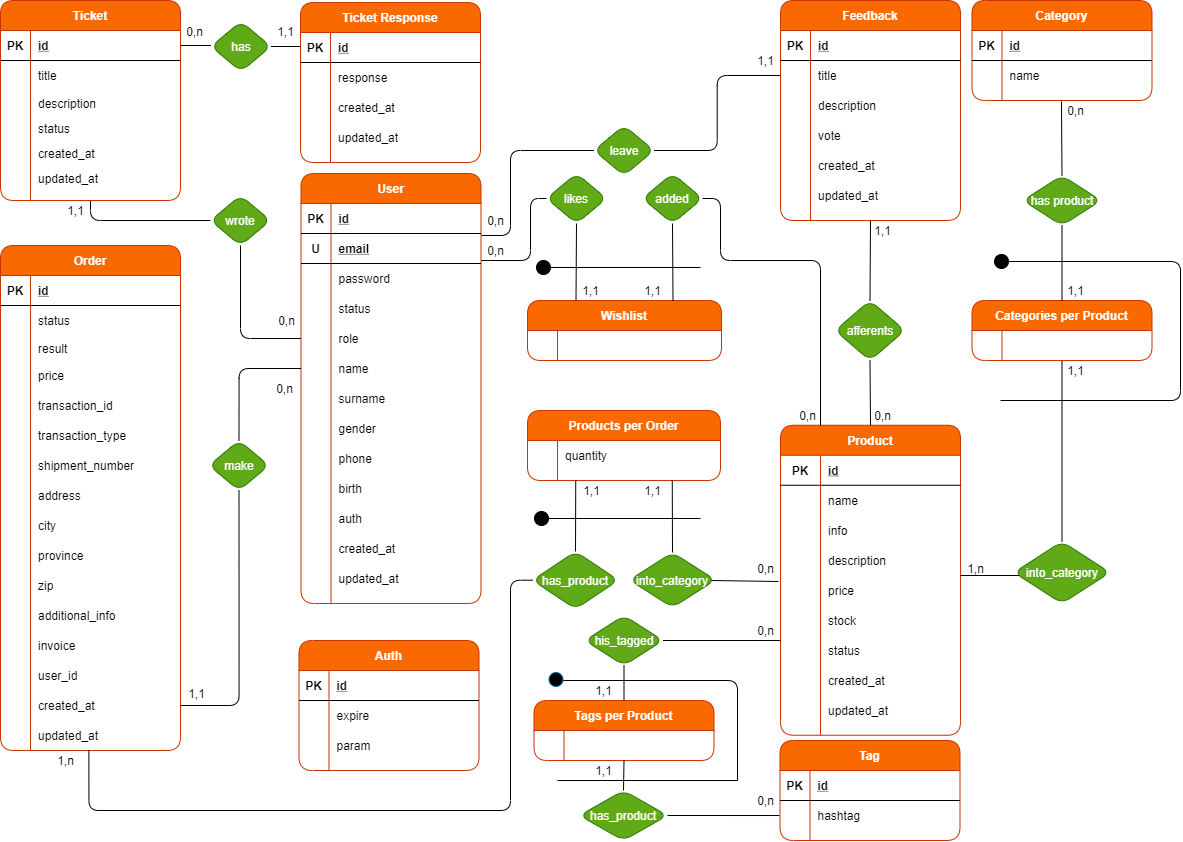


Figura 15

# Diagramma UML delle Classi

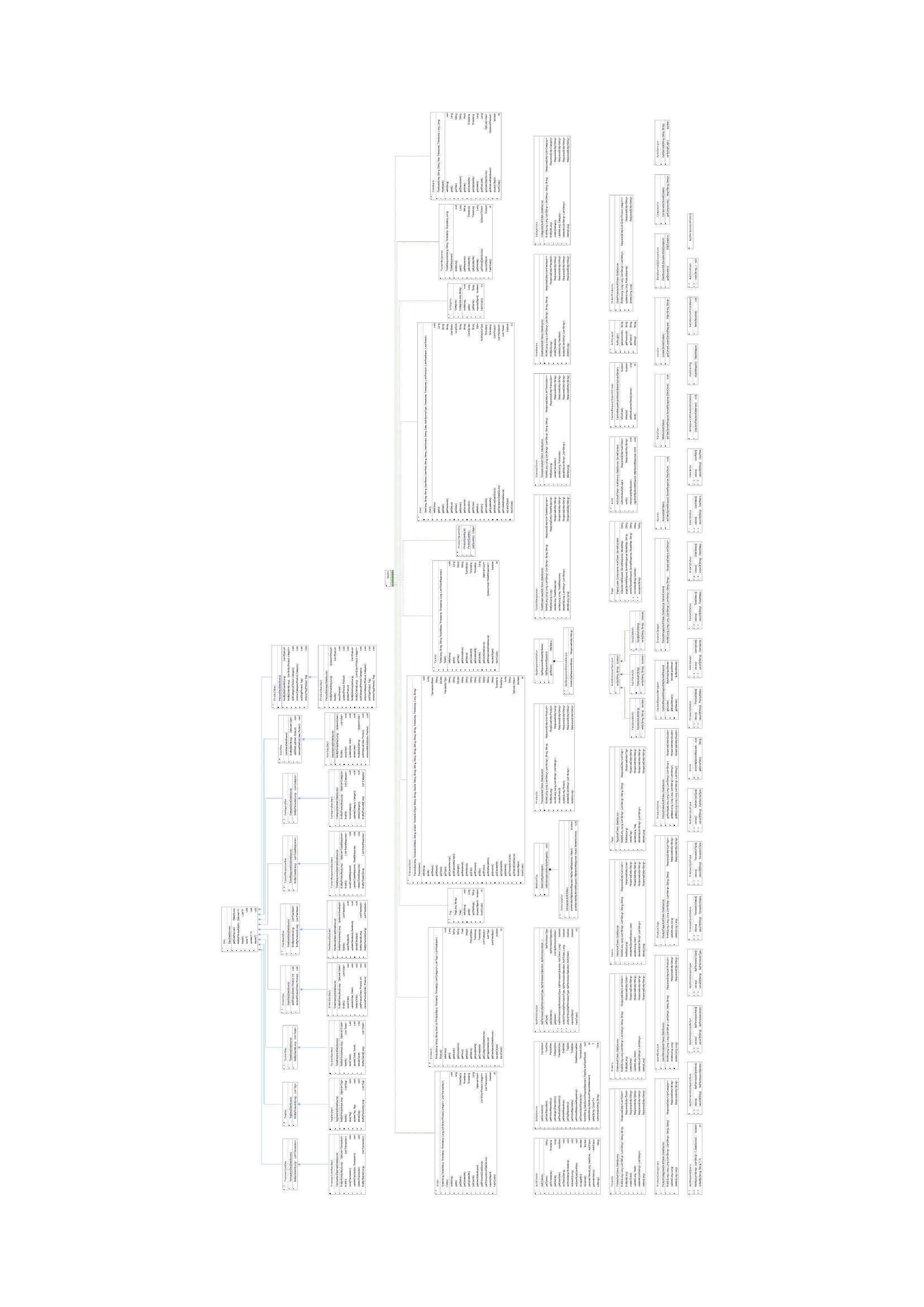


Figura 16

<https://github.com/bioagrisrls/public/raw/main/schema/uml.pdf>

# Tecnologie Utilizzate

## Back-end

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | PostgreSQL - Wikipedia | Felpe | Spalding 1876 | Google Italia (@googleitalia) | Twitter | Facebook - Wikipedia | Twitter - Wikipedia |

## Front-End

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | HTML5 - Wikipedia | JavaScript - Wikipedia | Sass: Brand Guidelines |  |  |

# Indice

## Figure

[Figura 1 37](#_Toc61905943)

[Figura 2 38](#_Toc61905944)

[Figura 3 38](#_Toc61905945)

[Figura 4 39](#_Toc61905946)

[Figura 5 39](#_Toc61905947)

[Figura 6 40](#_Toc61905948)

[Figura 7 41](#_Toc61905949)

[Figura 8 42](#_Toc61905950)

[Figura 9 43](#_Toc61905951)

[Figura 10 44](#_Toc61905952)

[Figura 11 45](#_Toc61905953)

[Figura 12 46](#_Toc61905954)

[Figura 13 59](#_Toc61905955)

[Figura 14 60](#_Toc61905956)

[Figura 15 64](#_Toc61905957)

[Figura 16 65](#_Toc61905958)

[Figura 17 67](#_Toc61905959)

[Figura 18 67](#_Toc61905960)