



UNIVERSITÀ  
DELLA CALABRIA

DIPARTIMENTO DI **MATEMATICA  
E INFORMATICA**



**bioagri**<sub>s.r.l.s.</sub>

# PROGETTO DI WEB COMPUTING

A.A. 2020/2021

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.

Matteo Perfidio – 200632  
Antonino Natale – 200705  
Davide Crisafulli – 195097

# SOMMARIO

Introduzione .....	2
Descrizione .....	3
API .....	3
Create .....	4
Read .....	4
Update.....	5
Delete .....	6
Auth .....	7
Services.....	7
Servizi di pagamento .....	7
Servizi di accesso .....	7
UI Framework.....	8
Navigazione .....	8
Supporto multilingua.....	8
Componenti .....	9
Gestore dei temi.....	10
Schema E-R.....	11
Diagramma UML delle Classi.....	12
Tecnologie Utilizzate .....	13
Back-end.....	13
Front-End.....	13
Indice .....	14
Figure .....	14

# 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 multiplatforma di un'applicazione web orientata allo **shopping online** e all'ingresso della stessa nel mercato digitale globale.

La piattaforma offre ai clienti, acquisiti e potenziali, tutte le informazioni e i consigli necessari sia nella fase di pre-vendita che di post-vendita.

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 architetetturalmente indipendenti tra loro: back-end e front-end; messi in comunicazione tramite API. L'intento è quello di creare un applicativo multiplatforma, RESTful e flessibile all'evoluzione della stessa.



FIGURA 1

## 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.

Ogni risorsa all'interno del server è raggiungibile, con i dovuti permessi e controlli, mediante un set di API comune a tutti i Model, per implementare un'interfaccia CRUD.

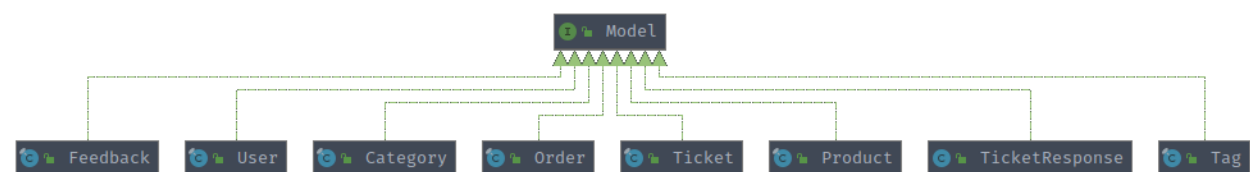


FIGURA 2

## CREATE

<b>POST</b>	<u>/api/resource</u>
-------------	----------------------

Crea una risorsa all'interno del database.

## CORPO DELLA RICHIESTA

```
1. {  
2.   "id": ...,  
3.   "name": "..."  
4. }
```

## RISPOSTA

<i>Codice</i>	<i>Descrizione</i>
201	Risorsa creata con successo
4xx	Si è verificato un errore nella creazione della risorsa

## READ

<b>GET</b>	<u>/api/resource</u>
<b>GET</b>	<u>/api/resource/{id}</u>

Ottiene una lista o una singola risorsa.

## PARAMETRI

<i>Nome</i>	<i>Descrizione</i>
<i>skip</i>	Salta i primi N elementi
<i>limit</i>	Limita il risultato ad N elementi
<i>filter-by</i>	Filtra il risultato per K chiave

<i>filter-val</i>	Filtra il risultato per V valore di K chiave
<i>sorted-by</i>	Ordina il risultato per K chiave
<i>order</i>	Metodologia di ordinamento

## RISPOSTA

<i>Codice</i>	<i>Descrizione</i>
200	Collezione o singolo elemento richiesto
4xx	Si è verificato un errore nell'ottenimento della risorsa

## ESEMPIO

Ottiene una lista dei primi cinque prodotti che contengono la stringa "acqua".

```
1. GET /api/products?filter-by=name&filter-val=(?:.*)(acqua)(?:.*)&limit=5
```

## UPDATE

<b>PUT</b>	<u>/api/resource/{id}</u>
------------	---------------------------

Aggiorna una risorsa all'interno del database.

## CORPO DELLA RICHIESTA

```
1. {
2.   "id": ...,
3.   "name": "..."
4. }
```

## RISPOSTA

Codice	Descrizione
200	Risorsa aggiornata con successo
4xx	Si è verificato un errore nell'aggiornamento della risorsa

## DELETE

<b>DELETE</b>	<u>/api/resource</u>
<b>DELETE</b>	<u>/api/resource/{id}</u>

Elimina una lista o una singola risorsa.

## PARAMETRI

Nome	Descrizione
<i>skip</i>	Salta i primi N elementi
<i>limit</i>	Limita il risultato ad N elementi
<i>filter-by</i>	Filtra il risultato per K chiave
<i>filter-val</i>	Filtra il risultato per V valore di K chiave

## RISPOSTA

Codice	Descrizione
204	Collezione o singolo elemento eliminato con successo
4xx	Si è verificato un errore nell'eliminazione della risorsa

## ESEMPIO

Elimina tutti i prodotti che contengono la stringa "acqua".

```
1. DELETE /api/products?filter-by=name&filter-val=(?:.*)(acqua)(?:.*)
```

## 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.

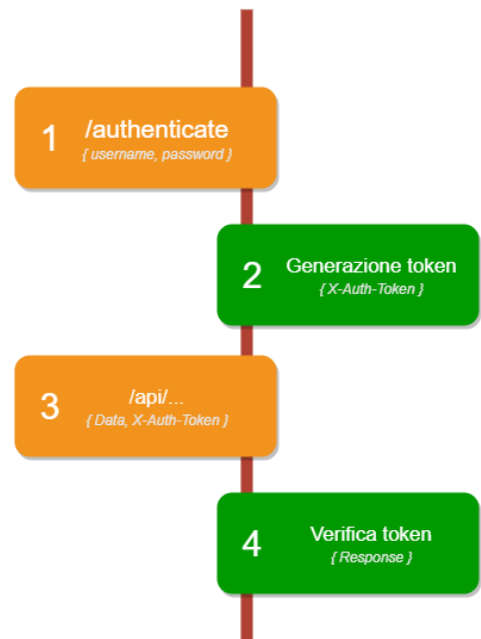


FIGURA 3

## SERVICES

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

### SERVIZI DI PAGAMENTO

- PayPal

### SERVIZI DI ACCESSO

- Google
- Facebook
- Twitter



## UI FRAMEWORK

Framework per lo sviluppo di applicazioni web, sviluppato interamente da noi, con l'esigenza di gestire tutti gli aspetti di un front-end modello **FAT-client**.

Il framework si compone di quattro elementi essenziali:

- Navigazione
- Componenti
- Gestore dei temi
- Supporto MULTI-lingua

## NAVIGAZIONE

Gestione della navigazione tra le pagine, in modo asincrono e senza la necessità di un aggiornamento completo della pagina, mediante l'utilizzo delle **History API** fornite dallo standard HTML5.

Attraverso tali funzionalità, fornite dai browser moderni, il framework precarica in background il contenuto della pagina successiva da visualizzare e ne sostituisce il contenuto dinamicamente, ove serve.

## SUPPORTO MULTILINGUA

Il framework, tramite la sua architettura di progettazione delle View, mette a disposizione dello sviluppatore una gestione dei contenuti testuali in molteplici lingue.

Definendo un **dizionario** chiave/valore in formato JSON all'interno del server è possibile, in base a determinate condizioni: come il contesto regionale del visitatore oppure le preferenze dell'utente stesso, cambiare dinamicamente tale dizionario, con i contenuti testuali della lingua scelta.

## COMPONENTI

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.

La View è costruita sulla base di un **template** che riduce lo sforzo per la creazione di contenuti dinamici, fornendo allo sviluppatore funzionalità come:

- Costrutti di programmazione: for, while, if, ecc...
- **Binding** dinamici ad una o due vie per la comunicazione dei dati tra Padre-figlio e viceversa.

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

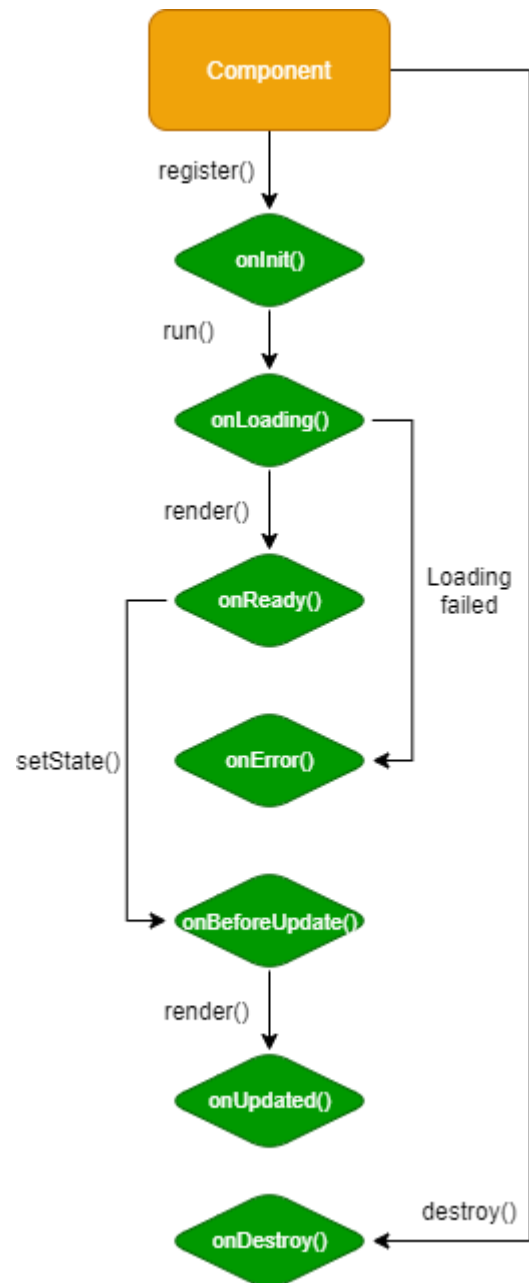


FIGURA 4

## GESTORE DEI TEMI

Entità per la gestione in tempo reale dei temi. Un tema è definito sulla base di uno schema globale di variabili **SASS**, che tramite la sostituzione di queste ultime va a costruire uno stile di **Bootstrap**.

Ogni variante possiede una versione *light* e *dark*, intercambiabile tramite un apposito **toggle** presente nella parte destra della *navbar*.



FIGURA 5

# SCHEMA E-R

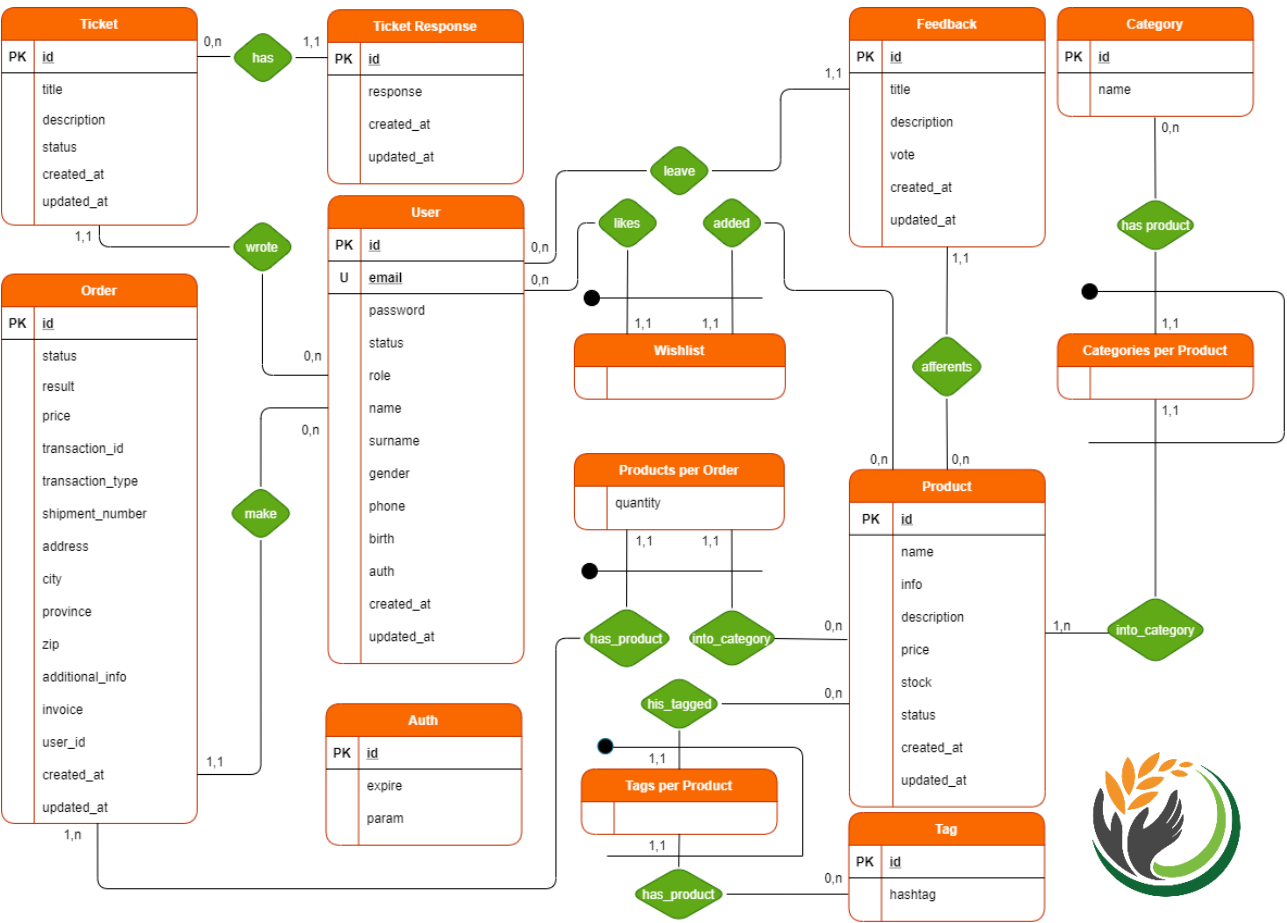


FIGURA 6

# DIAGRAMMA UML DELLE CLASSI



FIGURA 7

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

# TECNOLOGIE UTILIZZATE

## BACK-END

- *Java 15*
- *Spring Framework*
- *PostgreSQL*
- *PayPal Checkout SDK*
- *Google SDK*
- *Facebook SDK*
- *Twitter SDK*



## FRONT-END

- *Java Server Pages*
- *HTML5*
- *Javascript*
- *SASS*
- *Bootstrap 5*
- *Material Design Icons*



# INDICE

## FIGURE

Figura 1 .....	3
Figura 2.....	3
Figura 3.....	7
Figura 4.....	9
Figura 5.....	10
Figura 6.....	11
Figura 7.....	12