Gestionale Magazzino

# Indice

1 Indice 2

2 Introduzione 4

2.1 Informazioni sul progetto 4

2.2 Abstract 4

2.3 Scopo 4

2.3.1 Scopi didattici 4

2.3.2 Scopi operativi 4

3 Analisi 5

3.1 Analisi del dominio 5

3.2 Analisi e specifica dei requisiti 5

3.3 Use case 10

3.4 Pianificazione 11

3.5 Analisi dei mezzi 12

3.5.1 Software 12

3.5.2 Hardware 12

4 Progettazione 13

4.1 Design dell’architettura del sistema 13

4.2 Design dei dati e database 14

4.2.1 Diagramma ER 14

4.2.2 Descrizione delle Tabelle 16

4.2.3 Descrizione delle Relazioni 17

4.2.4 Procedure e Trigger 17

4.3 Design delle interfacce 18

4.3.1 Design iniziale 18

4.4 Design procedurale 22

4.4.1 Noleggio articoli 22

4.4.2 Chiusura noleggio 23

4.4.3 Sistema di notifica noleggio in scadenza 25

4.4.4 Sistema di notifica noleggio scaduto 25

5 Implementazione 26

5.1 Installazione MySQL 26

5.2 Installazione Nodejs 26

5.3 Installazione PM2 e script per l’auto deploy 27

5.4 Certificato TLS/SSL 28

5.5 Struttura applicativo 28

5.5.1 Database 29

5.5.2 Models 29

5.5.3 Controllers 30

5.5.4 Views 32

5.5.5 Middlewares 32

5.5.6 Routes 32

5.5.7 Tests 32

6 Test 33

6.1 Protocollo di test 33

6.2 Risultati test 41

6.3 Mancanze/limitazioni conosciute 41

7 Consuntivo 42

8 Conclusioni 44

8.1 Sviluppi futuri 44

8.2 Considerazioni personali 44

9 Bibliografia 45

9.1 Bibliografia per articoli di riviste: 45

9.2 Bibliografia per libri 45

9.3 Sitografia 45

10 Glossario 46

11 Indice delle figure 46

12 Allegati 46

# Introduzione

## Informazioni sul progetto

Allievi: Gioele Cappellari, Davide Branchi, Amir Kawsarani

Docente: Michel Palucci

Scuola: CPT Trevano SAMT I3

Data inizio: 12.01.2024

Data consegna: 03.05.2024

## Abstract

*Managing a medium-scale warehouse alone, without tools that simplify the work, can become a very tedious process and allow room for error. For this project we’ve decided to create a program that allows you to manage all of this automatically and intuitively. With it a person can choose whether to manage the warehouse all alone or distribute the load among several people. Without it, coordinating multiple people and updating a list of available equipment would be nearly impossible and would take a lot of time. Instead, with the use of the Warehouse Management System, these problems are solved and the time spent making sure that all the equipment is present is pretty much non existent.*

## Scopo

### Scopi didattici

Gli obiettivi principali di questo progetto sono farci lavorare usando la metodologia Agile e prepararsi per il progetto del quarto anno.

Invece riguardo alla creazione del progetto volevamo migliorare a interagire con le banche dati e integrarci un sistema di codici QR, il tutto utilizzando NodeJs per sviluppare il nostro applicativo.

### Scopi operativi

Senza un sistema di gestione del magazzino, gestirne uno di medie/grandi dimensioni diventa molto laborioso e duraturo. Perciò lo scopo del progetto è proprio quello di riuscire semplificare la gestione dei magazzini implementando un gestionale che è in grado di:

* Aggiornare Database tramite la lettura dei codici QR.
* Visualizzare una lista di oggetti e gestirla.
* Implementare un sistema a noleggi con date di scadenza.
* Implementare un sistema a utenti con diversi permessi.

# Analisi

## Analisi del dominio

Un’azienda di cinematografia ha un magazzino che contiene tutto il materiale che viene usato per registrare. Essa per gestirlo conta a mano che il materiale sia presente. Stessa cosa per quando viene portato via per fare dei progetti.

Essendo che la quantità di materiale da gestire è elevata e c’è solamente una persona a gestire il magazzino, queste azioni richiedono troppo tempo e sono troppo soggette a errori.

Per semplificare la gestione del magazzino verranno usati dei codici QR sul materiale per poterli memorizzare e scansionare in maniera veloce.

Per la parte di presa del materiale per i progetti viene introdotto un sistema che ti permette di vedere che cos’è disponibile e che gestisce in maniera automatica la parte di “noleggio”.

Esso verrà utilizzato principalmente dal gestore del magazzino. Esso dovrà leggere il manuale di utilizzo per capire il pieno funzionamento dell’applicativo e ridurre il margine di errore.

## Analisi e specifica dei requisiti

In base alle direttive del cliente, sono state date le seguenti specifiche per il software:

|  |  |
| --- | --- |
| **Req-001** | |
| **Nome** | Presenza database |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Deve essere presente un database dove verranno memorizzati tutti i dati riguardo al noleggio del materiale e le credenziali di accesso al sito |

|  |  |
| --- | --- |
| **Req-002** | |
| **Nome** | Login |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente necessita di un’interfaccia per accedere al sito.  Solo i dipendenti dell’azienda potranno fare l’accesso al sito. |
| **Sotto requisiti** | |
| **Req-001** | Si necessita la presenza di un database funzionante |

|  |  |
| --- | --- |
| **Req-003** | |
| **Nome** | Gestione articoli |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il gestore del magazzino deve poter gestire i prodotti.  Quando viene registrato un nuovo prodotto, viene generato un codice QR |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “gestore magazzino” |

|  |  |
| --- | --- |
| **Req-004** | |
| **Nome** | Visualizzazione articoli |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve poter visualizzare tutti gli articoli dell’azienda |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito |
| **Req-003** | Devono essere presenti degli articoli (già registrati) |

|  |  |
| --- | --- |
| **Req-005** | |
| **Nome** | Gestione categorie articoli |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il gestore del magazzino deve poter gestire le categorie dei prodotti |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “gestore magazzino” |

|  |  |
| --- | --- |
| **Req-006** | |
| **Nome** | Visualizzazione categorie |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il gestore del magazzino deve poter visualizzare tutte le categorie di prodotti |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “gestore magazzino” |
| **Req-005** | Devono essere presenti delle categorie (già registrate) |

|  |  |
| --- | --- |
| **Req-007** | |
| **Nome** | Visualizzazione informazioni singolo articolo |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve poter visualizzare le informazioni di un determinato articolo |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito |
| **Req-003** | Devono essere presenti degli articoli |

|  |  |
| --- | --- |
| **Req-008** | |
| **Nome** | Gestione articolo tramite QR |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve poter scannerizzare il codice QR per visualizzare la scheda dell’articolo |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito |
| **Req-003** | Devono essere presenti degli articoli |

|  |  |
| --- | --- |
| **Req-009** | |
| **Nome** | Gestione noleggi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’utente deve poter gestire i noleggi degli articoli |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito |
| **Req-003** | Devono essere presenti degli articoli |

|  |  |
| --- | --- |
| **Req-010** | |
| **Nome** | Stampa codice QR |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il gestore del magazzino deve poter stampare tramite etichettatrice i codici QR che fanno riferimento ad un articolo |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “gestore magazzino” |
| **Req-003** | Devono essere presenti degli articoli |
| **Req-006** | Visualizzazione QR dalle informazioni dell’articolo |

|  |  |
| --- | --- |
| **Req-011** | |
| **Nome** | Gestione inventario |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Il gestore del magazzino deve poter gestire l’inventario controllando che ci sia tutto il materiale |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “gestore magazzino” |
| **Req-003** | Devono essere presenti degli articoli |

|  |  |
| --- | --- |
| **Req-012** | |
| **Nome** | Gestione utenti sito |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’amministratore deve poter gestire gli utenti del sito, quindi deve poterli creare ed eliminare |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “amministratore” |

|  |  |
| --- | --- |
| **Req-013** | |
| **Nome** | Visualizzazione lista utenti |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | L’amministratore deve poter visualizzare la lista di tutti gli utenti presenti nell’applicativo |
| **Sotto requisiti** | |
| **Req-002** | L’utente deve aver fatto l’accesso al sito come “amministratore” |
| **Req-013** | Devono essere presenti degli utenti |

|  |  |
| --- | --- |
| **Req-014** | |
| **Nome** | Sistema di allerta per restituzione noleggio |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Note** | Deve esserci un sistema di notifica tramite email che deve avvisare l’utente quando sta per scadere il noleggio.  Se il noleggio è scaduto, deve rinviare l’email all’utente e deve allertare il gestore del magazzino. |

**Spiegazione elementi tabella dei requisiti:**

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

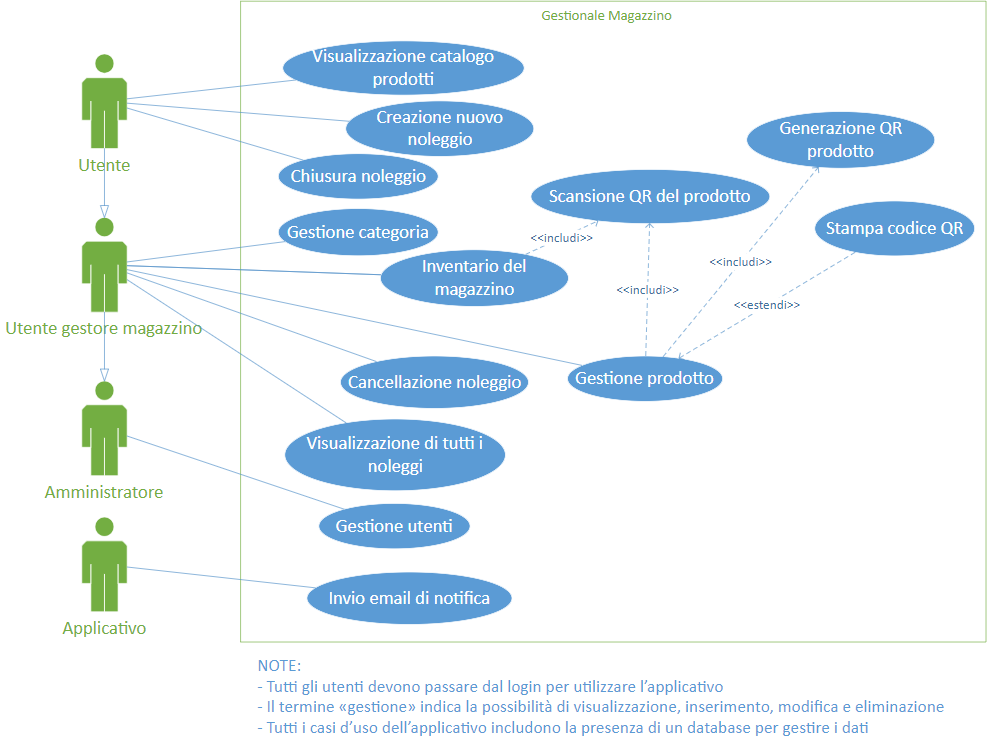


Figura 1 Use Case

Nell’applicativo l’attore principale sarà l’utente, che si specializza in utente gestore del magazzino e utente amministratore. Inoltre, sarà presente un altro attore che è l’applicativo stesso, e che avrà il compito di gestire il sistema di invio delle email di notifica per i noleggi in scadenza e scaduti.

Tutti gli utenti per poter utilizzare l’applicativo dovranno effettuare un login, i casi d’uso che rappresentano la differenza fra l’utente normale e il gestore sono la possibilità di gestire le categorie, effettuare l’inventario del magazzino, gestire i prodotti, visualizzare tutti i noleggi ed effettuare la chiusura forzata di un noleggio specifico.

L’utente amministratore invece avrà come possibilità in più la funzionalità di gestione degli utenti.

## Pianificazione

Per questo progetto abbiamo scelto di utilizzare una metodologia Agile, più precisamente abbiamo utilizzato il framework Scrumban.

Qui di seguito è presente il diagramma di Gantt con le attività pianificate in linea di massima e le relative tempistiche stimate:

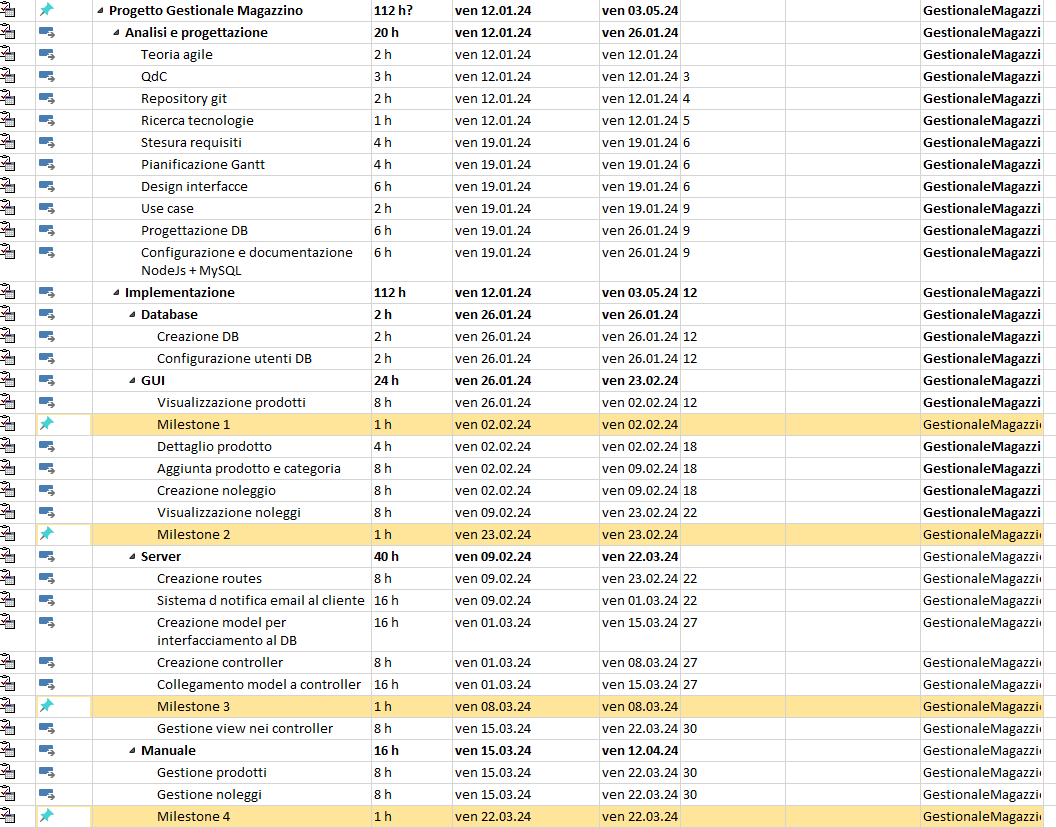


Figura 2 Diagramma di Gantt

Naturalmente la fase che in questo progetto richiederà più tempo sarà quella dell’implementazione. Abbiamo deciso di suddividere questa fase in quattro principali sottofasi: inizialmente abbiamo pianificato una fase di implementazione del database, poi abbiamo tutta la fase di creazione delle GUI, poi sarà presente l’implementazione del backend e infine la scrittura dei manuali per facilitare l’utilizzo da parte dell’utente finale.

## Analisi dei mezzi

### Software

* **MySQL 8.0.35**
* **Visual Studio Code 1.86.0**
* **NodeJS 20.11.0**
* **Npm 10.2.4**
* **Mozilla Firefox 103.0.1**
* **VirtualBox 7.0**: per hostare la macchina virtuale Ubuntu di sviluppo

I moduli di NodeJS che servono per il funzionamento dell’applicativo sono presenti in allegato nel file “moduli applicativo.docx”.

### Hardware

L’hardware utilizzato per lo sviluppo del software:

* **Una Webcam**: questa viene utilizzata per sviluppare e testare il funzionamento della lettura dei codici QR.
* **Un tablet**: esso serve per testare il responsive dell’applicativo e per simulare un possibile scenario dove il “gestore del magazzino” utilizza l’applicativo.
* **Un PC scolastico**: esso serve per sviluppare e testare l’applicativo. Il PC ha le seguenti specifiche:
  + Intel Core i7 7700
  + 16 GB di RAM
  + SSD 512 GB
  + Windows 10
* **I telefoni**: essi servono per testare il responsive dell’applicativo e per simulare un possibile scenario dove un “utente normale” utilizza l’applicativo e fa un noleggio.
* **Stampante**: quest’ultima serve per stampare i codici QR dei prodotti. Simula l’etichettatrice che dovrebbe stampare le etichette.

L’hardware il quale dovrà ospitare l’applicativo WEB:

* **Un server scolastico**: all’interno del server è hostata la macchina virtuale che utilizziamo per hostare l’applicativo e il database. La macchina virtuale ha le seguenti specifiche:
  + 2 GB di RAM
  + Ubuntu Server

# Progettazione

## Design dell’architettura del sistema

La struttura architetturale del nostro applicativo è molto semplice, infatti, essendo un'applicazione aziendale è destinata all'utilizzo esclusivo all'interno della rete dell’azienda. Per questo motivo non avremo bisogno di nessun hardware particolare.

L’infrastruttura è costituita da un singolo server, il quale si occupa di ospitare il database per la memorizzazione dei dati e l’applicativo web stesso.

Gli utenti dell’azienda potranno accedere all’applicativo utilizzando i PC aziendali, quindi attraverso la rete cablata interna, oppure tramite i dispositivi mobili o pc collegati alla rete wireless. Questi tipi di accesso sono stati progettati per facilitare la gestione dei noleggi per gli amministratori e i gestori; mentre per quanto riguarda gli altri utenti dell’azienda risulterà più comodo effettuare dei noleggi dato che possono utilizzare direttamente il telefono.

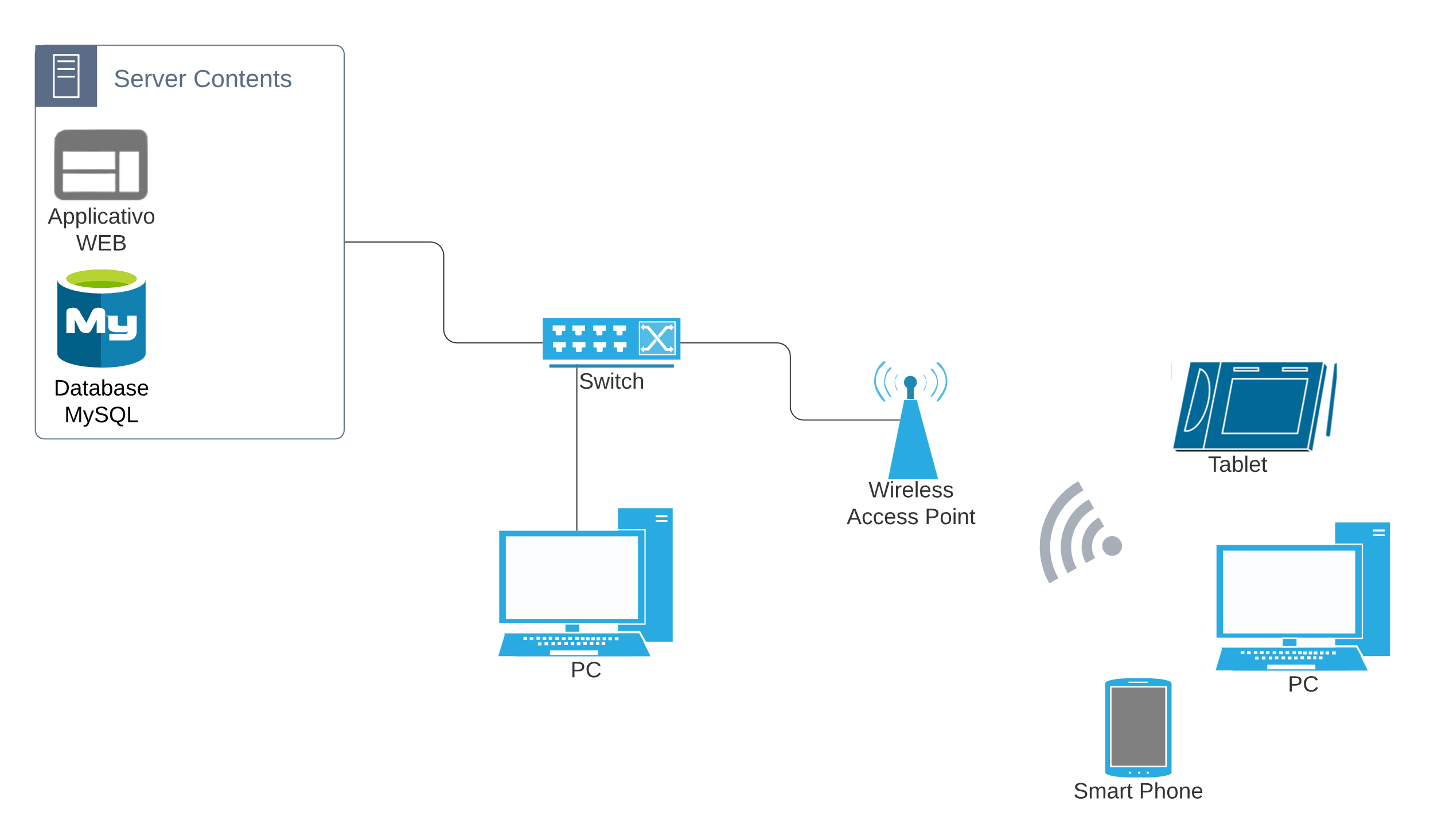


Figura 3 Architettura di sistema

## Design dei dati e database

### Diagramma ER

Prima versione:

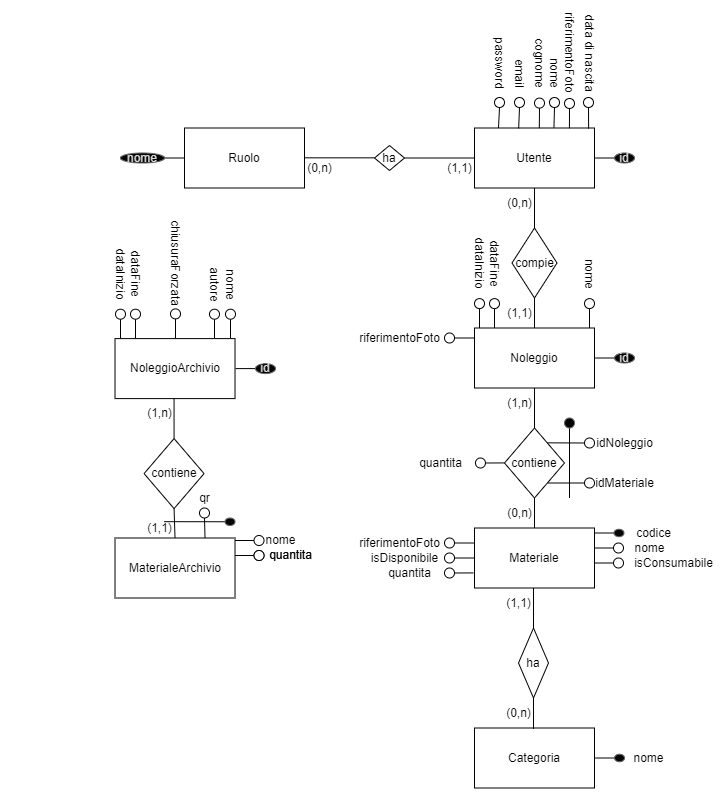


Figura 4 Prima versione diagramma ER

Dopo aver fatto la prima versione ci siamo accorti che non era necessario tenere la relazione tra “NoleggioArchivio” e “MaterialeArchivio” in base alle esigenze, perciò è stato modificato il diagramma ER e il database.

Seconda versione:

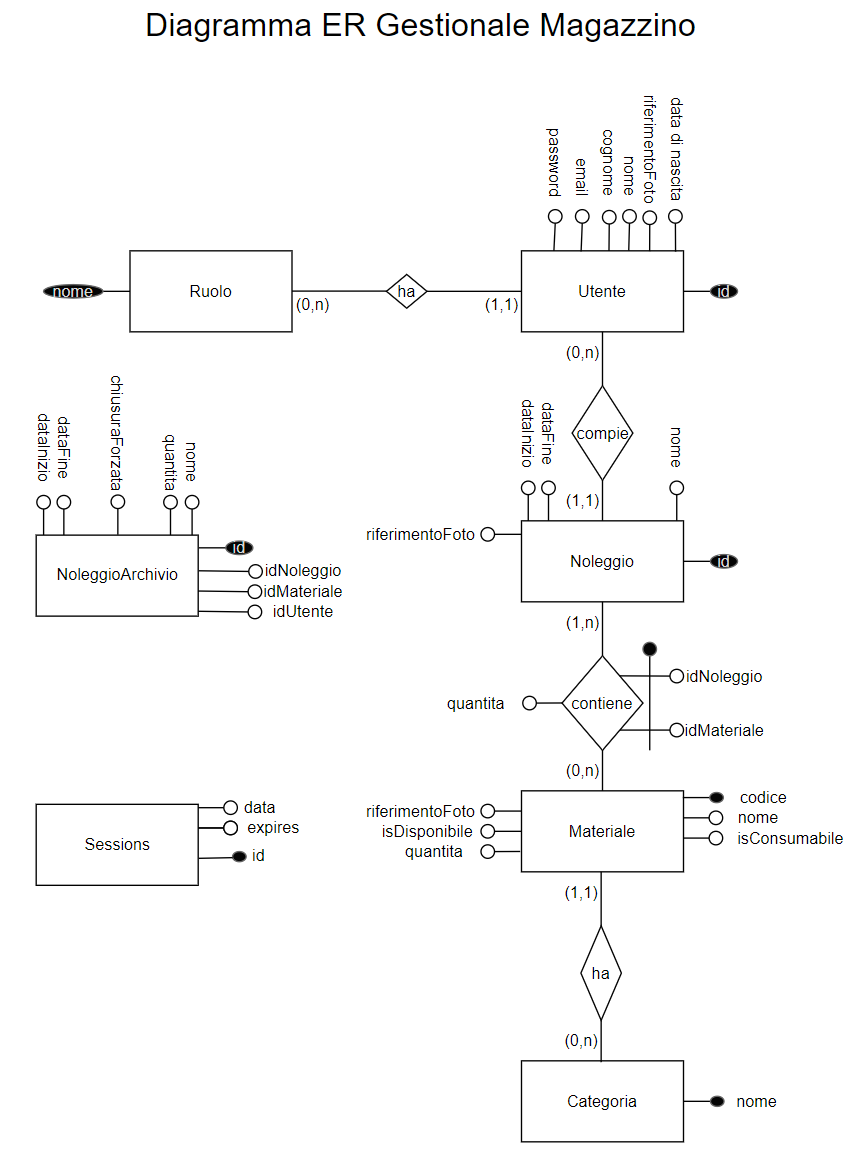


Figura 5 Seconda versione diagramma ER

### Descrizione delle Tabelle

**Ruolo**

* Contiene i ruoli degli utenti del sistema, come amministratori, gestori e utenti normali.
* Attributi: nome (chiave primaria).

**Sessions**

* Memorizza le sessioni degli utenti per il controllo dell'autenticazione.
* Attributi: id (chiave primaria), expires, data.

**Categoria**

* Rappresenta le categorie di materiali presenti nel magazzino.
* Attributi: nome (chiave primaria).

**Utente**

* Contiene informazioni sugli utenti del sistema.
* Attributi: id (chiave primaria), nome, cognome, riferimentoFoto, dataNascita, email, password, ruolo (chiave esterna verso la tabella Ruolo).

**Noleggio**

* Rappresenta i noleggi effettuati dagli utenti.
* Attributi: id (chiave primaria), nome, riferimentoFoto, dataInizio, dataFine, idUtente (chiave esterna verso la tabella Utente), chiusuraForzata.

**Materiale**

* Contiene informazioni sui materiali presenti nel magazzino.
* Attributi: codice (chiave primaria), nome, riferimentoFoto, quantita, isConsumabile, isDisponibile, categoria (chiave esterna verso la tabella Categoria).

**MaterialeNoleggio**

* Associa i materiali ai noleggi, indicando la quantità noleggiata.
* Attributi: idNoleggio (chiave esterna verso la tabella Noleggio), idMateriale (chiave esterna verso la tabella Materiale), quantita.

**Archivio**

* Conserva lo storico dei noleggi conclusi.
* Attributi: id (chiave primaria), nome, idNoleggio (valori presi dalla tabella ponte tra Noleggio e Materiale), idMateriale (valori presi dalla tabella ponte tra Noleggio e Materiale), idUtente, dataInizio, dataFine, quantita, chiusuraForzata.
* Chiusura forzata tiene conto se il noleggio è stato chiuso in maniera forzata (ha avuto delle perdite).
* Viene aggiornato usando una procedura perciò non è una vera e propria relazione.

### Descrizione delle Relazioni

La tabella:

* **Utente** ha una relazione con la tabella Ruolo attraverso l'attributo ruolo, che indica il ruolo dell'utente nel sistema.
* **Noleggio** è collegata alla tabella Utente tramite l'attributo idUtente, identificando l'utente che ha effettuato il noleggio.
* **Materiale** è collegata alla tabella Categoria tramite l'attributo categoria, specificando a quale categoria appartiene il materiale.
* **MaterialeNoleggio** collega i noleggi ai materiali noleggiati attraverso gli attributi idNoleggio e idMateriale, con una relazione molti a molti che indica che uno stesso materiale può essere noleggiato in più noleggi e un noleggio può contenere più materiali.
* **Archivio** registra i dettagli dei noleggi conclusi e archiviati, collegandoli tramite gli attributi idNoleggio e idMateriale alle rispettive tabelle di origine.

### Procedure e Trigger

**archivio\_update\_noleggio:**

Questo trigger viene attivato prima della cancellazione di una riga dalla tabella noleggio.

Prima della cancellazione, il trigger esegue la procedura descritta sopra per ciascuna riga nella tabella materialeNoleggio associata al noleggio che sta per essere eliminato.

L'uso di questo trigger assicura che, ogni volta che un noleggio viene eliminato, i dettagli dei materiali noleggiati vengano archiviati nella tabella archivio prima della cancellazione effettiva del noleggio stesso.

Ciò garantisce che lo storico dei noleggi e dei materiali noleggiati sia conservato anche dopo che i noleggi vengono cancellati dal sistema.

## Design delle interfacce

### Design iniziale

Per iniziare, abbiamo deciso di progettare le interfacce da mobile perché sono più semplici da fare ed è molto più intuitivo passare da un’interfaccia mobile a desktop. Poi rispetta di più le nostre esigenze di sviluppare un’applicazione comoda a livello di utente finale che utilizzerà principalmente il telefono.

#### Inventario

Queste è il design che era stato pattuito all’inizio. Una volta scelta la sezione inventario si vedrà una tabella con tutti i prodotti presenti nel magazzino e le rispettive informazioni. Se si clicca il pulsante di aggiunta di un nuovo prodotto, si deve aprire la GUI dove si inseriscono i dati del nuovo prodotto.

Mentre se si clicca sul bottone per visualizzare il dettaglio del prodotto, deve aprire una nuova interfaccia.

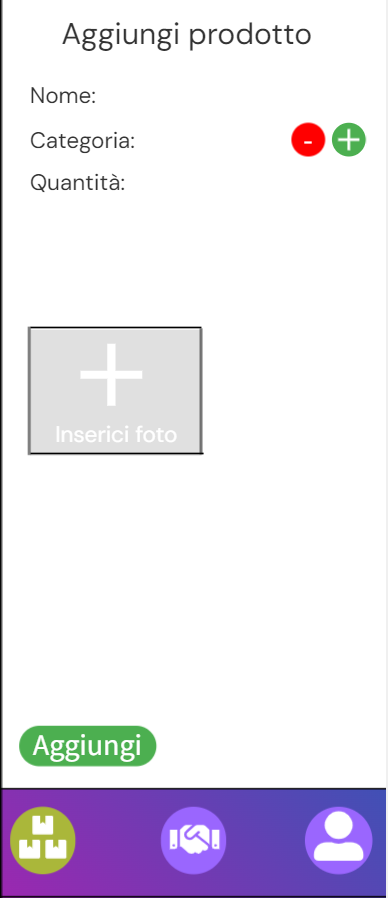
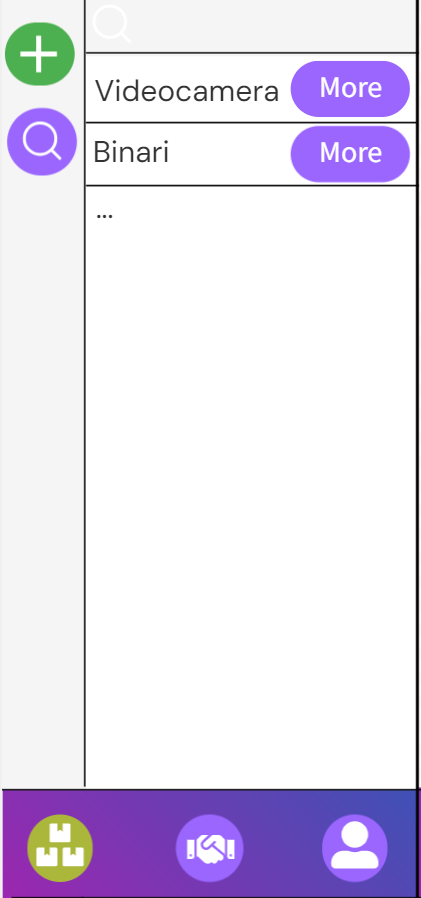


Figura 6 Design interfacce prodotti

#### Noleggi

Il design delle interfacce dei noleggi è simile a quello dell’inventario.

Quando si entra nella sezione dei noleggi, si ripresenterà la tabella simile a quella dell’inventario, solo che questa volta si vedranno tutti i noleggi.

Mentre se si clicca il bottone per vedere i dettagli dei noleggi si aprirà un'altra interfaccia che mostra tutti i dettagli del noleggio compresi i prodotti coinvolti in esso.

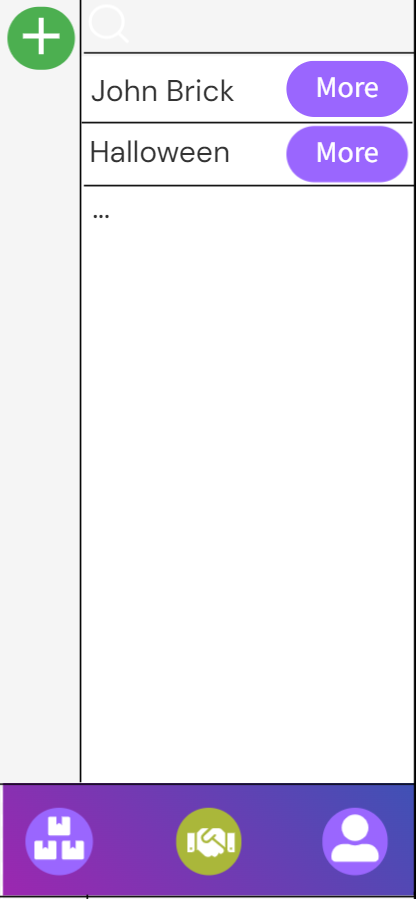


Figura 7 Desgin interfacce visualizzazione noleggi

Le altre due funzionalità sono quelle di “Aggiunta noleggio” e “Ritorna noleggio”.

L’interfaccia di aggiunta noleggio è abbastanza simile a quella dell’aggiunta di un prodotto, infatti ha i campi per inserire i dati necessari al noleggio e in più ha la funzionalità della scannerizzazione dei prodotti che si vogliono noleggiare.

L’interfaccia del ritorno di un noleggio invece è abbastanza semplice perché ha solo la funzionalità di scannerizzare gli oggetti del noleggio.

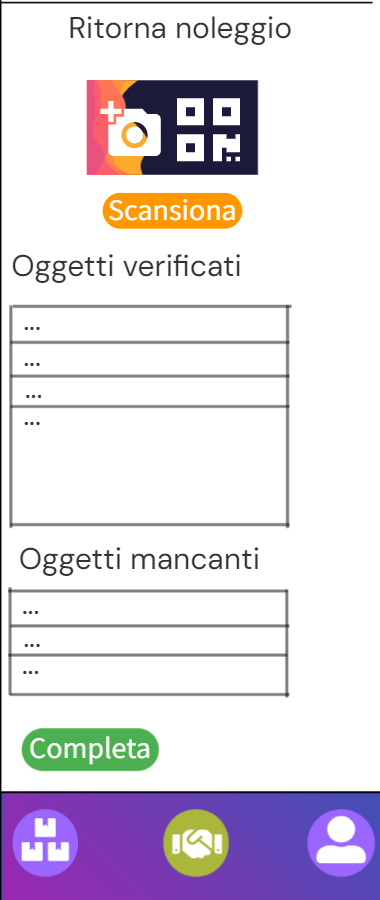


Figura 8 Design interfacce aggiunta e ritorno noleggio

#### Utente

Per il design delle interfacce dell’utente sono state riciclate quelle dell’inventario.

Infatti come si può vedere dalle immagini seguenti, si ripresenta la medesima tabella presente anche in inventario.

Quando si clicca il bottone per vedere i dettagli si aprirà un’interfaccia che mostrerà i dettagli dell’utente con la lista dei noleggi aperti.

Mentre se si clicca il pulsante per aggiungere un utente si aprirà una nuova interfaccia per l’aggiunta dell’utente.

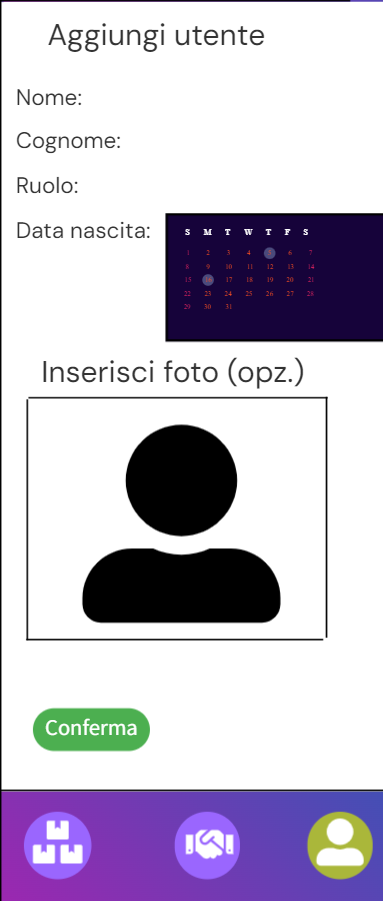
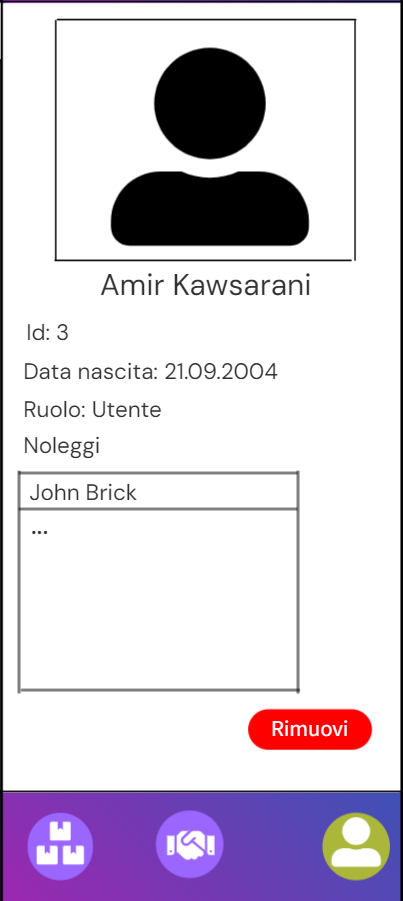
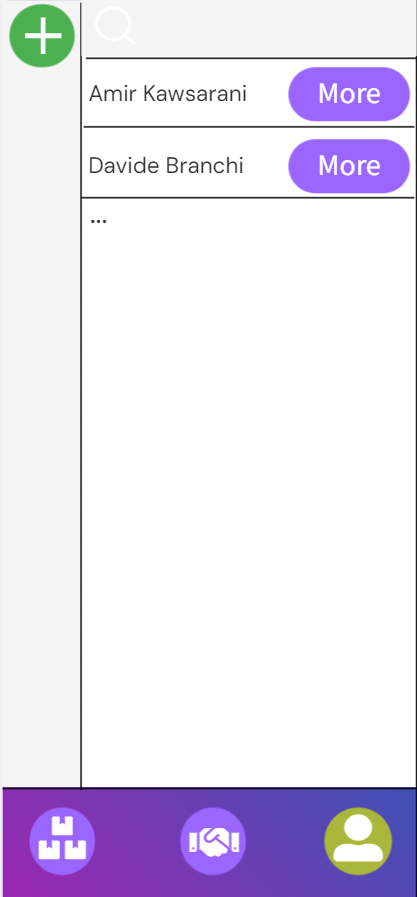


Figura 9 Design interfacce gestione utente

## Design procedurale

### Noleggio articoli

Il diagramma di flusso mostra il processo per noleggiare gli articoli. Per effettuare un noleggio, l'utente deve semplicemente scansionare tutti i codici QR presenti sugli articoli che desidera noleggiare. Nel frattempo, l'applicativo web si occuperà automaticamente del resto. Ogni volta che viene effettuata una scansione, l'applicativo aggiungerà l'oggetto al nuovo noleggio. Solo quando l'utente avrà terminato di scansionare tutti i codici QR, l'applicativo salverà nel database il nuovo noleggio.

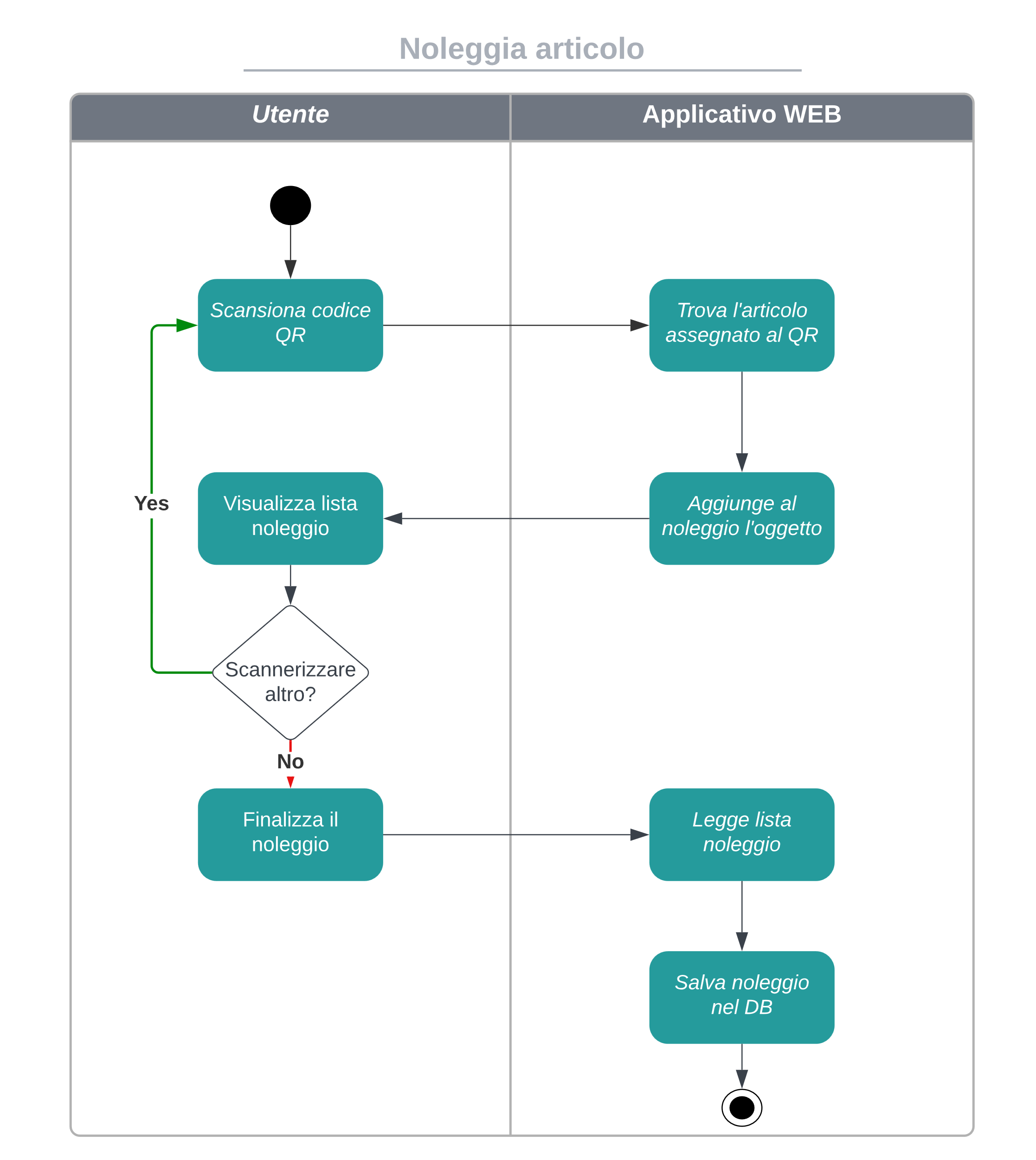


Figura 10 Diagramma di flusso noleggio articoli

### Chiusura noleggio

Il diagramma di flusso mostra il processo per chiudere un noleggio. Quando un utente deve restituire del materiale, deve prima selezionare il noleggio precedentemente creato, quindi cliccare sul pulsante "Ritorno materiale". Successivamente, deve scansionare tutti i codici QR presenti sul materiale preso dal magazzino. Dopo aver completato la scansione dei codici, deve cliccare sul pulsante "Chiusura noleggio". A questo punto, viene effettuato un controllo per verificare se tutto il materiale precedentemente preso è stato restituito. Se il materiale è al completo, il noleggio viene chiuso. Se manca del materiale, si presume che qualcosa sia stato perso. L’applicativo, quindi, verifica se l'utente che sta chiudendo il noleggio è un "gestore magazzino". In tal caso, il gestore può chiudere il noleggio tramite una chiusura forzata. Mentre se l'utente è un utente normale, viene richiesto di scansionare i codici QR non ancora scansionati. Nel caso l'utente desideri comunque chiudere il noleggio, deve contattare un gestore.

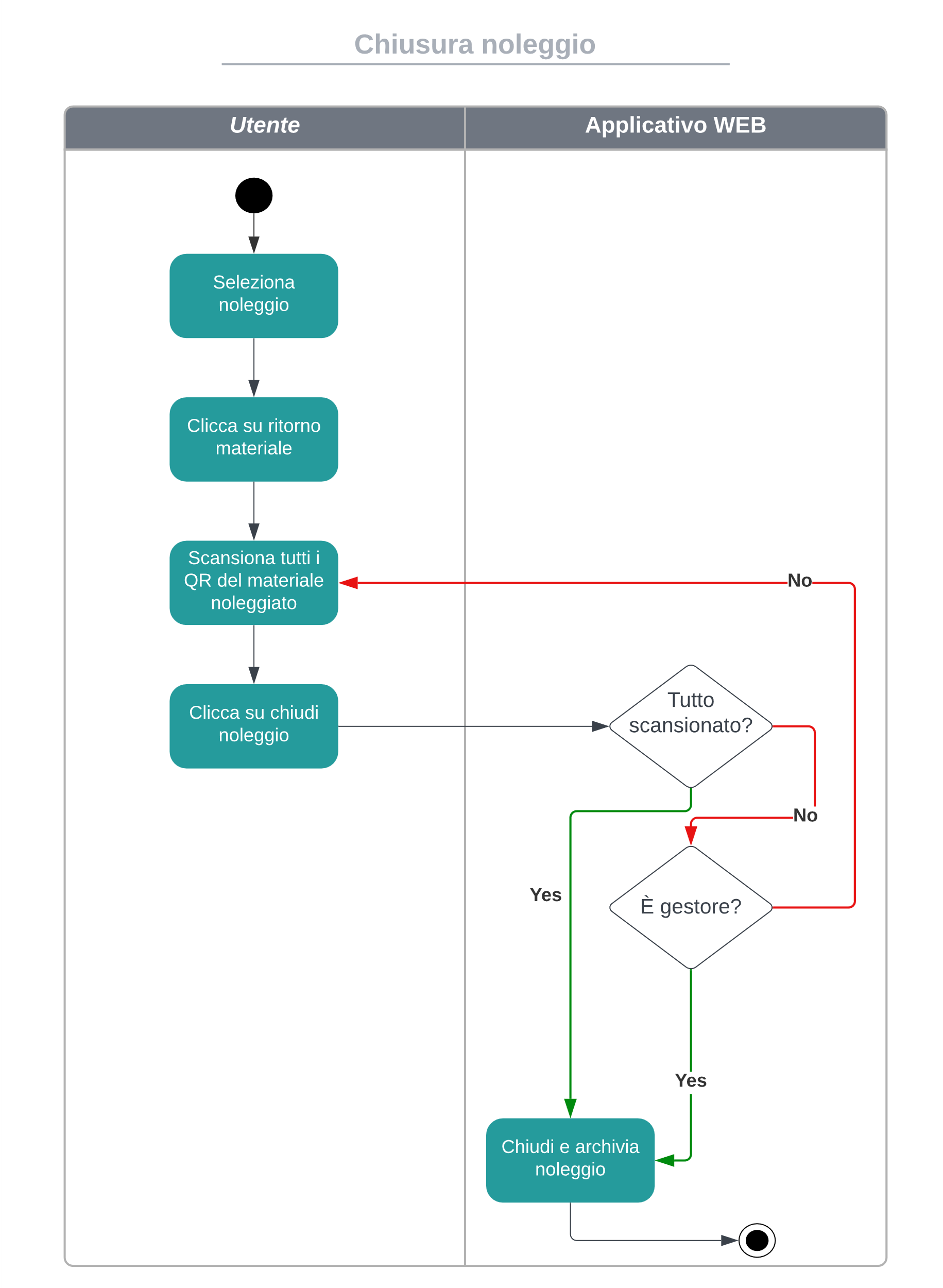


Figura 11 Diagramma di flusso chiusura noleggio

### Sistema di notifica noleggio in scadenza

Il diagramma di flusso mostra il processo del sistema di notifica per i noleggi che sono prossimi alla scadenza.

L’applicativo WEB verifica i noleggi che sono aperti e che sono a 24 ore dalla scadenza.

A questi utenti viene mandata un’email di notifica che li avverte della scadenza del loro noleggio.

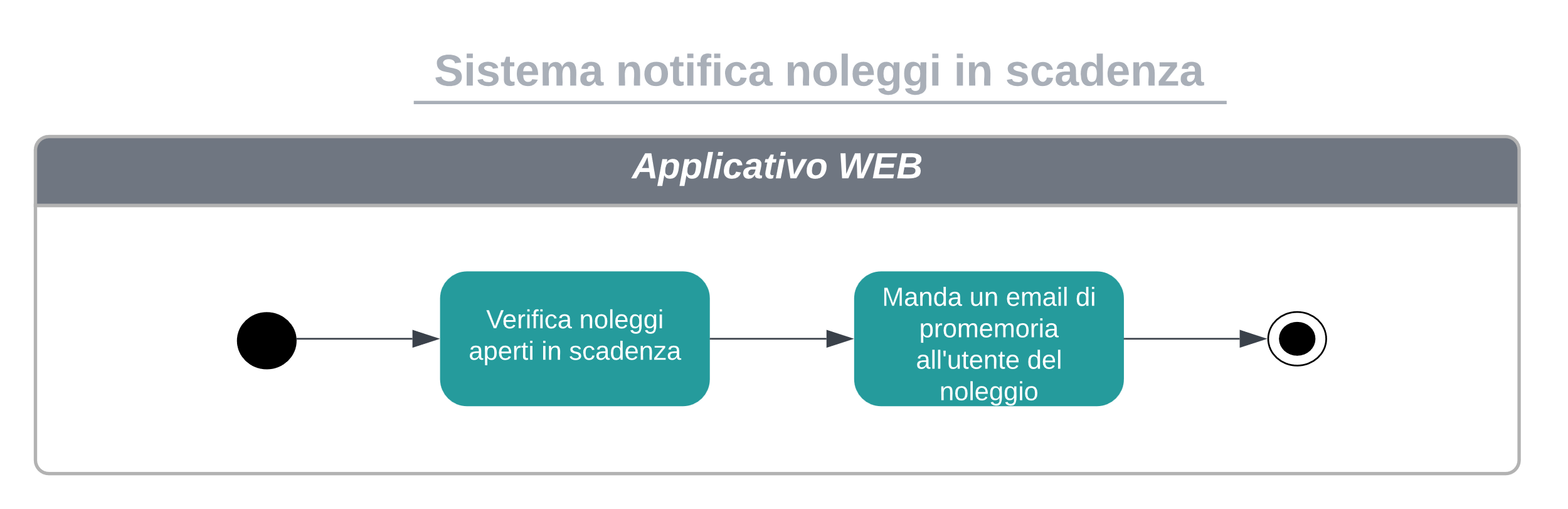


Figura 12 Diagramma di flusso sistema di notifica noleggio in scadenza

### Sistema di notifica noleggio scaduto

Il diagramma di flusso mostra il processo del sistema di notifica per i noleggi che sono scaduti.

L’applicativo WEB verifica i noleggi che sono aperti e che sono scaduti. A questi utenti viene inviata un’email che li avverte che il loro noleggio è scaduto e che devono riconsegnare il materiale. In più verrà mandata anche un email di allerta al gestore del magazzino.

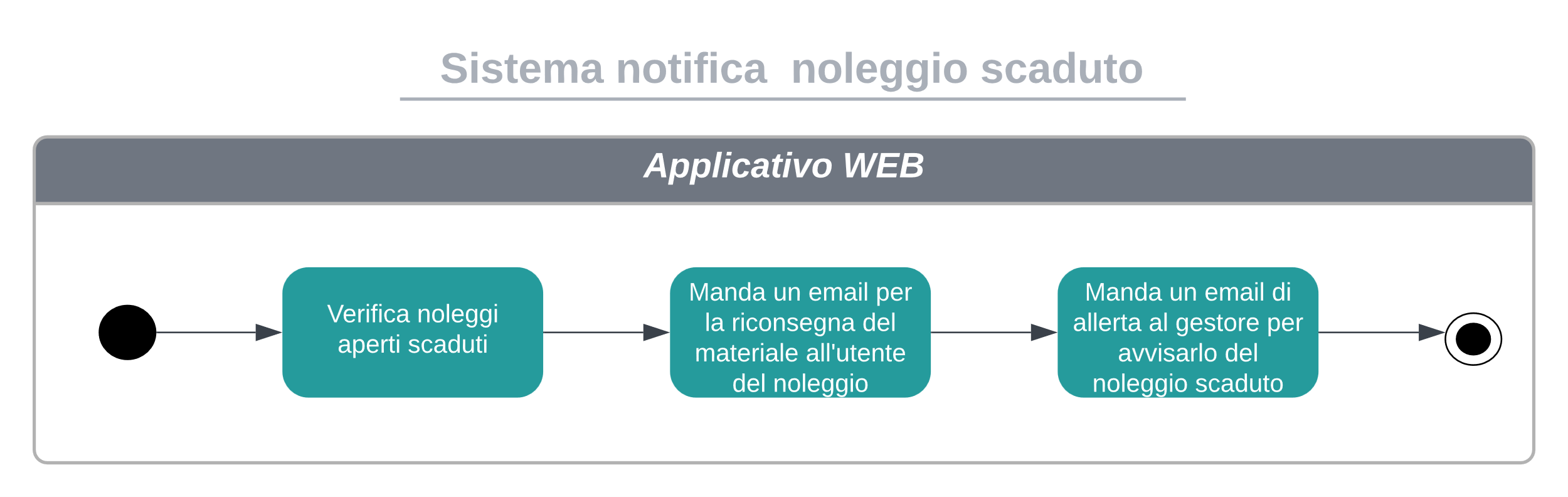


Figura 13 Diagramma di flusso sistema di notifica noleggio scaduto

# Implementazione

## Installazione MySQL

Sul server abbiamo deciso di installare il database MySQL per gestire tutti i dati dell’applicativo.

La versione installata è la 8.0.35.

I comandi eseguiti per l’installazione sono stati i seguenti:

Figura 14 Comandi installazione MySQL

sudo apt update

sudo apt install mysql-server

sudo mysql

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'tmp\_pass';

exit

sudo mysql\_secure\_installation

n

y

y

y

y

## Installazione Nodejs

Per il funzionamento del nostro applicativo WEB è stato utilizzato Nodejs.

La versione installata è la LTS corrente, ovvero la 20.11.0.

I seguenti comandi servono per installare curl, Nodejs e NPM:

Figura 15 Comandi installazione Nodejs

sudo apt install curl

curl -sL [https://deb.nodesource.com/setup\_20.x -o nodesource\_setup.sh](https://deb.nodesource.com/setup_20.x%20-o%20nodesource_setup.sh)

sudo bash ./nodesource\_setup.sh

sudo apt-get install nodejs -y

## Installazione PM2 e script per l’auto deploy

Per gestire il processo dell’applicazione Nodejs abbiamo installato PM2 che è un process manager per gestire gli applicativi.

In questo modo ogni volta che il server subirà ad esempio un riavvio, l’applicativo tornerà online automaticamente.

Ecco i comandi utilizzati per l’installazione e la configurazione:

Figura 16 Comandi installazione PM2

sudo npm install pm2@latest -g

sudo adduser app-user

sudo chown -R app-user:app-user /opt

su app-user

pm2 startup systemd

sudo env PATH=$PATH:/usr/bin /usr/lib/node\_modules/pm2/bin/pm2 startup systemd -u app-user --hp /home/app-user

sudo shutdown -r 0

sudo service pm2-app-user start

Nella cartella /opt/scripts è presente uno script auto\_deploy.py che controlla se su GitHub ci sono state modifiche alla repository, successivamente esegue un pull delle modifiche e riavvia il processo dell’app.

Figura 17 Script auto\_deploy.py

#!/usr/bin/python3

import os

import subprocess

os.chdir("/opt/GestionaleMagazzino")

git\_process = subprocess.Popen(["git", "fetch", "--dry-run"], stdout=subprocess.PIPE, stderr=subprocess.PIPE)

git\_fetch\_result = git\_process.communicate()

if git\_fetch\_result != (b'', b''):

    subprocess.check\_output("git pull", shell=True)

    subprocess.check\_output("pm2 restart app", shell=True)

Infine per fare in modo che lo script venga eseguito continuamente, è stato creato un CronJob che esegue lo script ogni minuto.

Figura 18 CronJob auto\_deploy.py

\* \* \* \* \* python3 /opt/scripts/auto\_deploy.py

## Certificato TLS/SSL

## Struttura applicativo

Per questo progetto abbiamo deciso di utilizzare il pattern MVC, la struttura di cartelle è la seguente:

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Figura 19 Struttura di cartelle dell'applicativo

* **database:** Questa cartella contiene i moduli necessari ad effettuare la connessione al database mysql
* **certs:** In questa cartella è presente il certificato ssl self-signed e la relativa chiave privata
* **middlewares:** Questa cartella contiene tutti i middleware, ovvero le funzioni da cui passa la richiesta prima di raggiungere il controller, un esempio è il middleware per verificare che l’utente sia autenticato
* **routes:** In questa cartella sono presenti i moduli che gestiscono tutte le routes dell’applicativo, integrando i middlewares e i controller
* **tests:** In questa cartella si trovano tutti gli unit test che vanno a testare il buon funzionamento dei models
* **views:** Questa cartella contiene tutte le pagine dell’applicativo scritte in EJS
* **controllers:** In questa cartella sono presenti tutti i controller, ovvero le funzioni che vanno a fare da tramite fra i models e le views
* **models:** Questa cartella contiene tutti i moduli che si occupano di andare a comunicare con il database e ritornare i dati ai controller
* **public:** In questa cartella sono contenute tutte le immagini, script JS, file CSS e librerie lato client, che devono essere accessibili dal web
* **node\_modules:** Questa cartella viene utilizzata per salvare tutti i moduli scaricati tramite NPM da cui l’applicativo dipende

### Database

Per effettuare la connessione al database mysql abbiamo utilizzato la libreria *mysql2*, scaricato tramite NPM. Nel modulo che abbiamo creato si va a creare un pool di connessioni al database e viene utilizzata la funzione *promise()* per fare in modo che si possa andare a chiamare il database in maniera asincrona.

const db = mysql.createPool({

host: process.env.DB\_HOST,

port: process.env.DB\_PORT,

user: process.env.DB\_USER,

password: process.env.DB\_PASS,

database: process.env.DB\_SCHEMA

}).promise();

Figura 20 Connessione al database

### Models

Per le entità più importanti presenti nel database (materiale, utente, noleggio, categoria, archivio), abbiamo creato delle classi molto semplici che contengono tutti gli attributi necessari.

class Noleggio{

id;

nome;

riferimentoFoto;

dataInizio;

dataFine;

idUtente;

chiusuraForzata;

constructor(id, nome, riferimentoFoto, dataInizio, dataFine, idUtente, chiusuraForzata){

this.id = id;

this.nome = nome;

this.riferimentoFoto = riferimentoFoto;

this.dataInizio = dataInizio;

this.dataFine = dataFine;

this.idUtente = idUtente;

this.chiusuraForzata = chiusuraForzata;

}

}

Figura 21 Classe Noleggio (model)

Tutti gli altri models sono simili all’esempio soprastante della classe “Noleggio”, in ogni model sono presenti gli attributi che rispecchiano le colonne presenti nelle tabelle del database.

Per andare a prendere i dati dal database sono stati creati dei moduli “mapper”, per ognuna delle entità più importanti citate in precedenza. Questi moduli contengono tutte le funzioni più utilizzate per prendere i dati dal database, qui di seguito sono presenti le funzioni per prendere tutti i noleggi e tutti i materiali di un noleggio.

async function getAll(){

const [result] = await db.query("SELECT \* FROM noleggio");

let noleggi = [];

for(let item of result){

noleggi.push(new Noleggio(item.id, item.nome, item.riferimentoFoto, item.dataInizio, item.dataFine, item.idUtente, item.chiusuraForzata));

}

return noleggi;

}

Figura 22 funzione getAll() - NoleggioMapper

async function getMaterialeOfNoleggio(idNoleggio){

const materialeMapper = require("./materialeMapper");

const [result] = await db.query("SELECT \* FROM materialeNoleggio WHERE idNoleggio=?", [idNoleggio]);

//Array bidimensionale con i materiali e le loro relative quantità

//es: [[materiale, quantita], [materiale, quantita] ecc...]

let materiali = [];

for(let item of result){

let m = await materialeMapper.getByCodice(item.idMateriale);

materiali.push([m, item.quantita]);

}

return materiali;

}

Figura 23 funzione getMaterialeOfNoleggio() - NoleggioMapper

### Controllers

Anche i controllers sono presenti per le entità più importanti, quest’ultimi fanno da tramite fra i models (che hanno il compito di andare a prendere i dati), e le views da mostrare all’utente. Inoltre nei controller vengono santificati tutti i dati e vengono anche fatti tutti i controlli per verificare che i dati inseriti dall’utente siano validi. Qui di seguito è presente il codice del controller per effettuare il login e del controller che mostra la schermata con tutti i noleggi.

async function login(req, res){

let {email, password} = req.body;

email = sanitizer.sanitizeInput(email);

password = sanitizer.sanitizeInput(password);

if (!sanitizer.validateEmail(email)){

let message = "Inserire un formato di email valido";

return res.status(401).render("login/index.ejs", { displayError: true, message: message });

}

const user = await userMapper.getByEmail(email);

if (!user){

return res.status(401).render("login/index.ejs", { displayError: true, message: "Inserire delle credenziali valide" });

}

const passwordEqual = await bcrypt.compare(password, user.password);

if (!passwordEqual){

return res.status(401).render("login/index.ejs", { displayError: true, message: "Inserire delle credenziali valide" });

}

req.session.user = user;

req.session.save(function() {

return res.status(200).redirect("/home");

});

}

Figura 24 funzione login() - loginController

Figura 25 funzione showAll() - noleggiController

async function showAll(req, res){

let noleggi = await noleggioMapper.getAllByDate();

noleggi = await noleggioMapper.changeIdUtenteToNome(noleggi);

return res.status(200).render("noleggio/noleggi.ejs", {noleggi: noleggi, session: req.session});

}

### Views

Le views vengono renderizzate lato server grazie ad EJS, alle views vengono passate tramite i controllers le variabili contenenti gli oggetti da mostrare. In ogni view inoltre viene passata anche la variabile di sessione (*req.body.session*) in questo modo si può accedere dalla view ad esempio alle informazioni dell’utente e permette di utilizzare la sidebar in maniera automatica.

#### Header

Per semplificare la creazione delle pagine abbiamo creato un template di header che viene richiamato sempre da ogni pagina all’interno dell’applicativo. Questa view rappresenta

la parte in comune di tutte le pagine, ovvero l’head.

Figura 26 Template EJS header

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta name="description" content="Progetto terzo anno secondo semestre">

<meta name="author" content="Gioele Cappellari, Amir Kawsarani, Davide Branchi">

<link rel="icon" href="/img/fav/favicon.ico" type="image/ico">

<!-- bootstrap -->

<link rel="stylesheet" href="/bootstrap/css/bootstrap.min.css">

<script src="/bootstrap/js/bootstrap.min.js"></script>

<!-- boxicons -->

<link rel="stylesheet" href="/boxicons/css/boxicons.css">

<script src="/boxicons/dist/boxicons.js"></script>

<!-- navbar -->

<link rel="stylesheet" href="/css/navbar.css">

Oltre ai meta tag, all’interno dell’header abbiamo importato anche la favicon e le librerie grafiche come bootstrap che ci servono in tutte le views.

Per implementare la parte dell’header nelle view bisogna aggiungere la seguente parte di codice:

Figura 27 Implementazione header nelle view

<head>

<%- include('../\_templates/header'); -%>

<title>Gestionale Magazzino - Dashboard</title>

<link rel="stylesheet" href="/css/dashboard/dashboard.css">

</head>

#### Sidebar

Per semplificare la navigazione abbiamo deciso di mettere ogni funzionalità dell’applicativo sulla sidebar. Essa usa un font importato per creare le icone, è responsive ed ha delle animazioni che gli permettono di uscire ed entrare se premuta.

Abbiamo anche implementato un sistema che sfrutta EJS per capire dove ci si trova, infatti colora di bianco in automatico la funzionalità selezionata.

Grazie a questo sistema è stato possibile isolarla come template in un file a parte.

### Middlewares

### Routes

### Tests

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-001  REQ-002 | **Nome** | Accesso utente |
| **Descrizione** | Si proverà a fare l’accesso all’applicativo con un utente normale. | | |
| **Prerequisiti** | Pagina di login completata | | |
| **Procedura** | 1. Cercare sul proprio browser l’indirizzo ip del server che ospita l’applicativo 2. Inserire le credenziali di login e cliccare sul pulsante login | | |
| **Risultati attesi** | Ci si dovrebbe ritrovare nella pagina home dell’applicativo con le funzionalità limitate a quelle di un utente normale. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-002  REQ-002 | **Nome** | Accesso utente con credenziali non corrette |
| **Descrizione** | Si proverà a fare l’accesso all’applicativo con un utente ma con delle credenziali non corrette o addirittura non esistenti. | | |
| **Prerequisiti** | Pagina di login completata | | |
| **Procedura** | 1. Cercare sul proprio browser l’indirizzo ip del server che ospita l’applicativo 2. Inserire delle credenziali di login errate e cliccare sul pulsante di login | | |
| **Risultati attesi** | Dovrebbe mostrare un messaggio di errore che le credenziali non sono corrette. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-003  REQ-002 | **Nome** | Accesso come gestore magazzino |
| **Descrizione** | Si proverà a fare l’accesso al sito come “gestore magazzino”. | | |
| **Prerequisiti** | Pagina di login completata | | |
| **Procedura** | 1. Cercare sul proprio browser l’indirizzo ip del server che ospita l’applicativo 2. Inserire delle credenziali di login di un “gestore magazzino” e cliccare sul pulsante di login | | |
| **Risultati attesi** | Ci si dovrebbe ritrovare nella pagina home dell’applicativo con le funzionalità di un “gestore magazzino”. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-004  REQ-002 | **Nome** | Accesso come amministratore |
| **Descrizione** | Si proverà a fare l’accesso al sito come “amministratore”. | | |
| **Prerequisiti** | Pagina di login completata | | |
| **Procedura** | 1. Cercare sul proprio browser l’indirizzo ip del server che ospita l’applicativo 2. Inserire delle credenziali di login di un “amministratore” e cliccare sul pulsante login | | |
| **Risultati attesi** | Ci si dovrebbe ritrovare nella pagina home dell’applicativo con le funzionalità di un “amministratore”. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-005  REQ-004 | **Nome** | Visualizzazione articoli |
| **Descrizione** | Si proverà ad andare a visualizzare gli articoli presenti nel database. | | |
| **Prerequisiti** | Pagina di visualizzazione articoli completata e devono essere presenti degli articoli nel DB | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Cliccare nel menu sull’icona per visualizzare gli articoli | | |
| **Risultati attesi** | Si dovrebbero visualizzare degli articoli. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-006  REQ-003 | **Nome** | Registrazione di un articolo |
| **Descrizione** | Si proverà a registrare un articolo nel database. | | |
| **Prerequisiti** | Pagina di registrazione articolo e pagina di visualizzazione articolo completate | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “gestore magazzino” 2. Cliccare nel menu sull’icona per aggiungere un articolo 3. Compilare i campi necessari e cliccare sul pulsante per aggiungere l’articolo 4. Cliccare nel menu sull’icona per visualizzare gli articoli | | |
| **Risultati attesi** | Come ultimo articolo si dovrebbe visualizzare quello appena registrato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-007  REQ-003 | **Nome** | Eliminazione di un articolo |
| **Descrizione** | Si proverà ad eliminare un articolo presente nel database. | | |
| **Prerequisiti** | Pagina di visualizzazione articolo completata | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “gestore magazzino” 2. Cliccare nel menu sull’icona per visualizzare gli articoli 3. Cliccare sul pulsante per eliminare l’articolo | | |
| **Risultati attesi** | L’articolo eliminato dovrebbe sparire dalla lista. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-008  REQ-006 | **Nome** | Visualizzazione categorie |
| **Descrizione** | Si proverà ad andare a visualizzare le categorie di prodotti presenti nel database. | | |
| **Prerequisiti** | Pagina visualizzazione categorie completata e devono essere presenti delle categorie di prodotti nel DB | | |
| **Procedura** | 1. Effettuare l’accesso come “gestore magazzino” 2. Cliccare nel menu sull’icona per visualizzare le categorie | | |
| **Risultati attesi** | Si dovrebbero visualizzare delle categorie. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-009  REQ-005 | **Nome** | Registrazione di una nuova categoria |
| **Descrizione** | Si proverà ad andare a registrare una nuova categoria di prodotti. | | |
| **Prerequisiti** | Pagina visualizzazione categorie e pagina di registrazione categorie completate. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “gestore magazzino” 2. Cliccare nel menu sull’icona per aggiungere una categoria 3. Compilare i campi necessari e cliccare sul pulsante per aggiungere la categoria 4. Cliccare nel menu sull’icona per visualizzare le categorie | | |
| **Risultati attesi** | Si dovrebbe visualizzare la nuova categoria appena registrata. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-010  REQ-005 | **Nome** | Eliminazione categoria prodotti |
| **Descrizione** | Si proverà ad eliminare una categoria. | | |
| **Prerequisiti** | Pagina visualizzazione categorie completata. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “gestore magazzino” 2. Cliccare nel menu sull’icona per visualizzare le categorie 3. Cliccare sul pulsante di eliminazione della categoria | | |
| **Risultati attesi** | Dovrebbe sparire dalla lista la categoria appena eliminata. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-011  REQ-007 | **Nome** | Visualizzazione informazioni singolo articolo |
| **Descrizione** | Si proverà ad andare a visualizzare le informazioni dell’articolo selezionato. | | |
| **Prerequisiti** | Pagina scheda articolo e pagina visualizzazione articolo completate. | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Cliccare nel menu sull’icona per visualizzare gli articoli 3. Cliccare su un articolo | | |
| **Risultati attesi** | Si dovrebbero visualizzare le informazioni sull’articolo. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-012  REQ-008 | **Nome** | Visualizzazione scheda articolo tramite QR |
| **Descrizione** | Si proverà ad andare a visualizzare la scheda dell’articolo scansionando il suo codice QR. | | |
| **Prerequisiti** | Applicativo web funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Cliccare nel menu sull’icona per scansionare un articolo 3. Scansionare il codice QR | | |
| **Risultati attesi** | Si dovrebbe visualizzare la scheda dell’articolo. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-013  REQ-009 | **Nome** | Noleggio articolo tramite codice QR |
| **Descrizione** | Si proverà ad andare a noleggiare un articolo scansionando il suo codice QR. | | |
| **Prerequisiti** | Applicativo web funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Cliccare nel menu sull’icona per scannerizzare l’articolo 3. Scannerizzare il codice QR presente sull’articolo 4. Cliccare sul pulsante noleggia | | |
| **Risultati attesi** | Si dovrebbe noleggiare l’articolo, quindi deve risultare noleggiato nel DB e nella lista dei noleggi. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-014  REQ-009 | **Nome** | Ritorno di un articolo tramite codice QR |
| **Descrizione** | Si proverà a ritornare l’articolo noleggiato scansionando il codice QR. | | |
| **Prerequisiti** | Applicativo web funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Cliccare nel menu sull’icona per ritornare l’articolo 3. Scannerizzare il codice QR presente sull’articolo 4. Cliccare sul pulsante di ritorno articolo | | |
| **Risultati attesi** | L’articolo dovrebbe sparire dai noleggi e lo si dovrebbe poter visualizzare nella lista di articoli presenti. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-015  REQ-010 | **Nome** | Stampa etichetta con codice QR articolo |
| **Descrizione** | Si proverà a stampare l’etichetta con il codice QR. | | |
| **Prerequisiti** | Applicativo web funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “gestore magazzino” 2. Cliccare nel menu sull’icona per visualizzare gli articoli 3. Cliccare su un articolo per visualizzarlo 4. Cliccare sul pulsante stampa QR 5. Scansionare l’etichetta appena stampata | | |
| **Risultati attesi** | Quando si scansiona l’etichetta stampata, si dovrebbe visualizzare l’articolo corrente. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-016  REQ-011 | **Nome** | Inventario articoli |
| **Descrizione** | Si proverà a fare il controllo dell’inventario. | | |
| **Prerequisiti** | Applicativo web funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “gestore magazzino” 2. Cliccare nel menu sull’icona per fare l’inventario 3. Scannerizzare tutti i codici degli articoli a parte uno | | |
| **Risultati attesi** | Dovrebbe dire quale articolo non è stato scannerizzato, quindi quale articolo manca. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-017  REQ-013 | **Nome** | Visualizzazione lista utenti |
| **Descrizione** | Si proverà ad andare a visualizzare tutti gli utenti registrati nell’applicativo. | | |
| **Prerequisiti** | Pagina visualizzazione utenti completata e presenti utenti nel DB. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “amministratore” 2. Cliccare nel menu sull’icona per visualizzare gli utenti | | |
| **Risultati attesi** | Si dovrebbero visualizzare gli utenti registrati nell’applicativo. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-018  REQ-012 | **Nome** | Registrazione di un nuovo utente |
| **Descrizione** | Si proverà a registrare un nuovo utente. | | |
| **Prerequisiti** | Pagina di visualizzazione utenti e pagina registrazione utenti completate. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “amministratore” 2. Cliccare nel menu sull’icona per registrare gli utenti 3. Compilare i campi necessari e cliccare sul pulsante per aggiungere l’utente 4. Spostarsi nella pagina di visualizzazione utenti | | |
| **Risultati attesi** | Si dovrebbe visualizzare l’utente appena registrato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-019  REQ-012 | **Nome** | Eliminazione utente |
| **Descrizione** | Si proverà ad eliminare un utente. | | |
| **Prerequisiti** | Pagina visualizzazione utenti completata e presenti utenti nel DB. | | |
| **Procedura** | 1. Effettuare l’accesso al sito come “amministratore” 2. Cliccare nel menu sull’icona per visualizzare gli utenti 3. Cliccare sul pulsante per eliminare l’utente | | |
| **Risultati attesi** | L’utente dovrebbe essere eliminato e quindi anche sparito dalla lista. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-020  REQ-014 | **Nome** | Allerta giorno prima della scadenza del noleggio |
| **Descrizione** | Si proverà a testare il sistema di allerta del giorno prima della restituzione del prodotto. | | |
| **Prerequisiti** | Applicativo web e sistema di allerta funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Noleggiare un articolo con scadenza a due giorni di distanza 3. Aspettare il giorno dopo e controllare le e-mail | | |
| **Risultati attesi** | Si dovrebbe ricevere un’e-mail che avvisa che il prodotto deve essere riconsegnato, i gestori e amministratori dovrebbero ricevere il report con nessun noleggio scaduto. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case**  **Riferimento** | TC-021  REQ-014 | **Nome** | Allerta giorno dopo alla scadenza del noleggio |
| **Descrizione** | Si proverà a testare il sistema di allerta il giorno dopo che il noleggio è scaduto | | |
| **Prerequisiti** | Applicativo web e sistema di allerta funzionante. | | |
| **Procedura** | 1. Effettuare l’accesso al sito con qualsiasi utente 2. Noleggiare un articolo con scadenza lo stesso giorno 3. Aspettare il giorno dopo e controllare le e-mail | | |
| **Risultati attesi** | Dovrebbe essere arrivata un’e-mail che avvisa che il prodotto deve essere riconsegnato e dovrebbe essere arrivata anche una email a tutti i gestori e amministratori con il report dei noleggi scaduti | | |

## Risultati test

La seguente tabella contiene i vari test case eseguiti all’applicativo e il loro esito finale.

Le comprove si trovano in allegato nel file: “Test case.docx”.

|  |  |  |
| --- | --- | --- |
| **Test Case** | **Risultato ottenuto** | **Stato** |
| TC-001 |  | Passato |
| TC-002 |  | Fallito |
| TC-003 |  |  |
| TC-004 |  |  |
| TC-005 |  |  |
| TC-006 |  |  |
| TC-007 |  |  |
| TC-008 |  |  |
| TC-009 |  |  |
| TC-010 |  |  |
| TC-011 |  |  |
| TC-012 |  |  |
| TC-013 |  |  |
| TC-014 |  |  |
| TC-015 |  |  |
| TC-016 |  |  |
| TC-017 |  |  |
| TC-018 |  |  |
| TC-019 |  |  |
| TC-020 |  |  |
| TC-021 |  |  |

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

Stile orizzontale



CONSUNTIVO

Figura Esempio di diagramma di Gantt consuntivo

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

1. URL del sito (se troppo lungo solo dominio, evt completo nel diario),
2. Eventuale titolo della pagina (in italico),
3. Data di consultazione (GG-MM-AAAA).

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Significato** |
| ASCII | American Standard Code for Information Interchange, codice per la codifica di caratteri. |
|  |  |
|  |  |
|  |  |
|  |  |
| ZFS | Zettabyte File System, filesystem opernsource a 128 bit con funzionalità avanzate. |

# Indice delle figure

[Figura 1 Esempio di diagramma di Gantt **Errore. Il segnalibro non è definito.**](#_Toc124775984)

[Figura 2 Esempio di diagramma di Gantt **Errore. Il segnalibro non è definito.**](#_Toc124775985)

[Figura 3 Diagramma ER Chen 14](#_Toc124775986)

[Figura 4 Diagramma ER Barker 15](#_Toc124775987)

[Figura 5 Esempio di diagramma di Gantt consuntivo 33](#_Toc124775988)

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …