

Piattaforma web per la raccolta differenziata

Applicazioni e Servizi Web

Sofia Tosi - 0001091387 {sofia.tosi4@studio.unibo.it}
Serafino Pandolfini - 0001094150 {serafino.pandolfini@studio.unibo.it}
Laura Leonardi - 0001094151 {laura.leonardi9@studio.unibo.it}

3 luglio 2023

Contents

1	Introduzione	1
2	Requisiti	2
3	Design	2
3.1	Mockup	3
3.2	Analisi Target Users	14
3.3	Storyboard	16
4	Tecnologie	27
4.1	frontend	28
4.2	schema db	29
4.3	backend	31
4.4	sicurezza	31
4.5	Gamification	32
5	Codice	32
5.1	interazione con Vuex	32
5.2	Gestione per la simulazione dei depositi	34
5.3	Richieste tramite Socket.io	35
6	Test	36
6.1	Euristiche di Nielsen	36
6.2	Test di usabilità	37
7	Deployment	37
7.1	Installazione	37
7.2	Messa in funzione	38
7.3	Simulazione	38
8	Conclusioni	38

1 Introduzione

Il progetto proposto consiste nella realizzazione di un'applicazione web chiamata Hè che permetta di gestire lo stato del riciclaggio dei rifiuti in Romagna e che permetta agli utenti registrati di monitorare la loro produzione di rifiuti. In particolare l'applicazione si focalizzerà sulla possibilità di gestire un area personale per ogni utente dove potrà visualizzare un report per ogni mese dove saranno indicati i cambiamenti nella produzione di rifiuti rispetto al mese precedente e la sua spesa mensile in rifiuti divisa per tipologie, una sezione grafici dove si

potranno osservare sia dati relativi al quantitativo di rifiuti prodotti che il loro valore in percentuale. In ogni sezione personale sarà anche presente una home riassuntiva dei contenuti riservati agli utenti registrati e una sezione dedicata ai premi che gli utenti potranno ricevere al fine di incentivare la raccolta differenziata. Nell'applicazione sarà anche presente una sezione disponibile a tutti dove saranno presenti dei grafici generali che utilizzeranno i dati di tutti gli utenti registrati e una classifica che premierà i comuni con gli utenti più virtuosi.

2 Requisiti

Descrizione delle caratteristiche e funzionalità che il sistema prevede.

I principali requisiti funzionali del sistema sono:

- **Autenticazione e registrazione utenti:** possibilità di accedere ad un'area privata per poter visualizzare informazioni personali, oltre alla presenza di un'area comune senza autenticazione
- **Statistiche e grafici:** possibilità per l'utente di visualizzare grafici a livello complessivo e personale, riguardanti mese e anno corrente, oltre al confronto tra questo mese e il precedente.
- **Notifiche:** sistema di notifiche al deposito rifiuti da parte dell'utente che lo avvisano dell'impatto economico di tale deposito
- **Ricompense utenti:** agli utenti più virtuosi saranno assegnate medaglie in base a svariati criteri, con l'intenzione di spingerli a riciclare maggiormente
- **Stato riempimento cassonetti:** possibilità per gli utenti di visualizzare lo stato dei cassonetti per facilitare la ricerca di un cassonetto disponibile
- **Classifica dei comuni:** classifica dei comuni più virtuosi, ha lo scopo di sfruttare il campanilismo tra comuni per incentivare gli utenti a migliorare la posizione del proprio comune

I requisiti non funzionali del sistema sono:

- **Usabilità:** rendere il sito facilmente utilizzabile da svariate tipologie di utente diverse
- **Estendibilità:** ottenere un sito facilmente estendibile con nuove funzionalità
- **Contenuto:** il sistema deve fornire informazioni utili ed interessanti a diverse categorie di utenti

3 Design

Design dell'architettura del sistema e delle interfacce utente.

3.1 Mockup

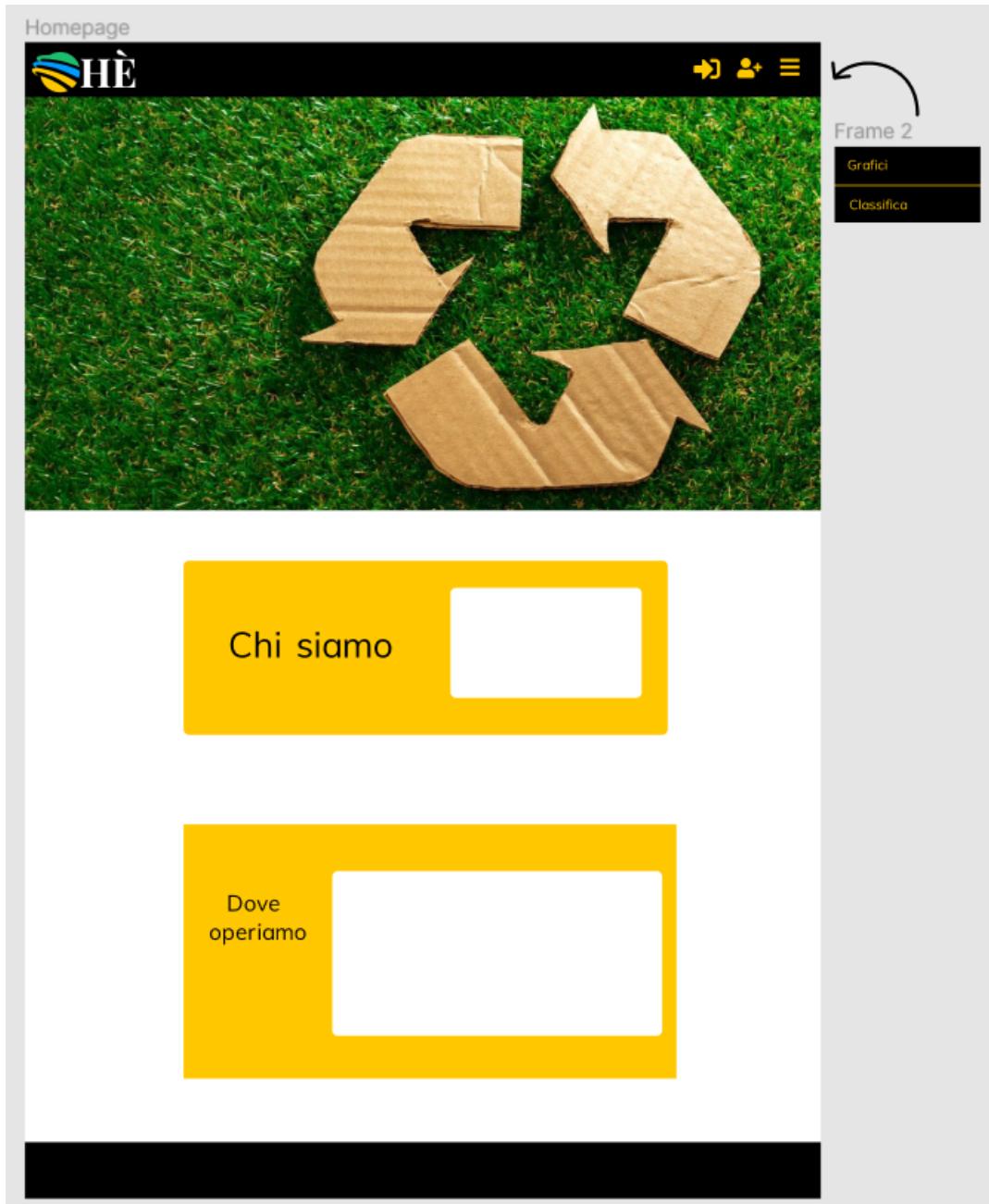


Figure 1: Mockup schermata home per Desktop



Figure 2: Mockup schermata home per i dispositivi mobile

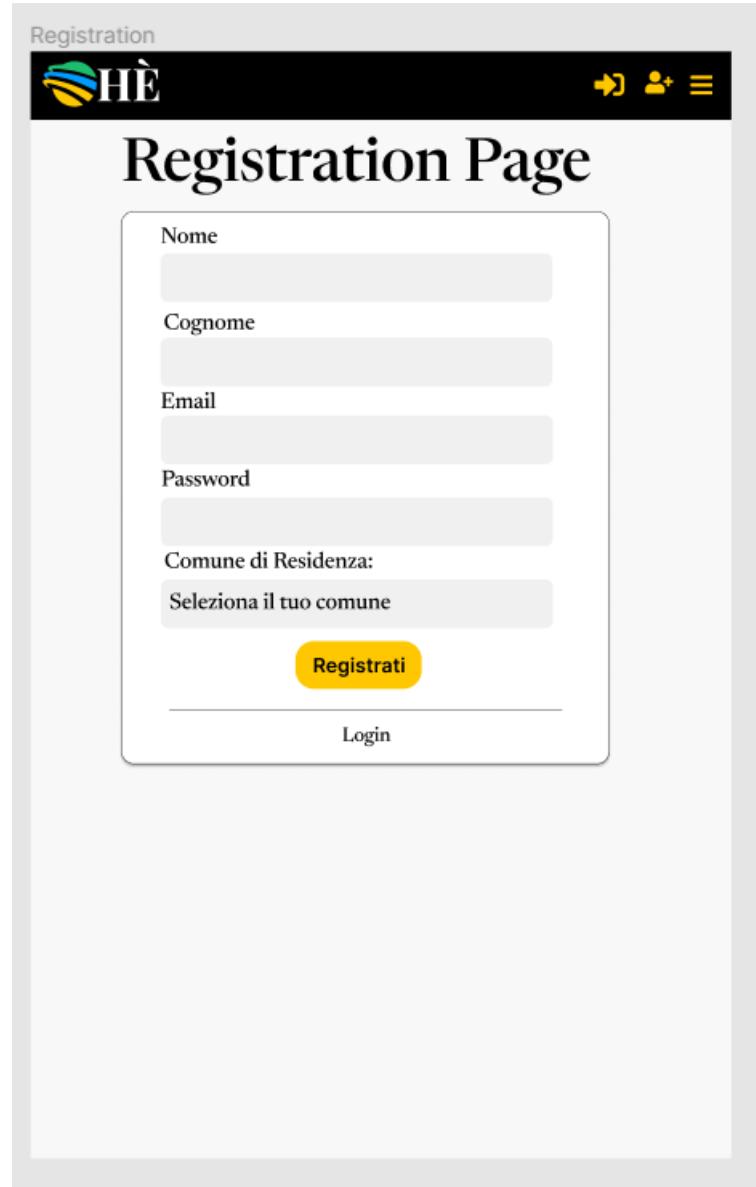


Figure 3: Mockup schermata di registrazione

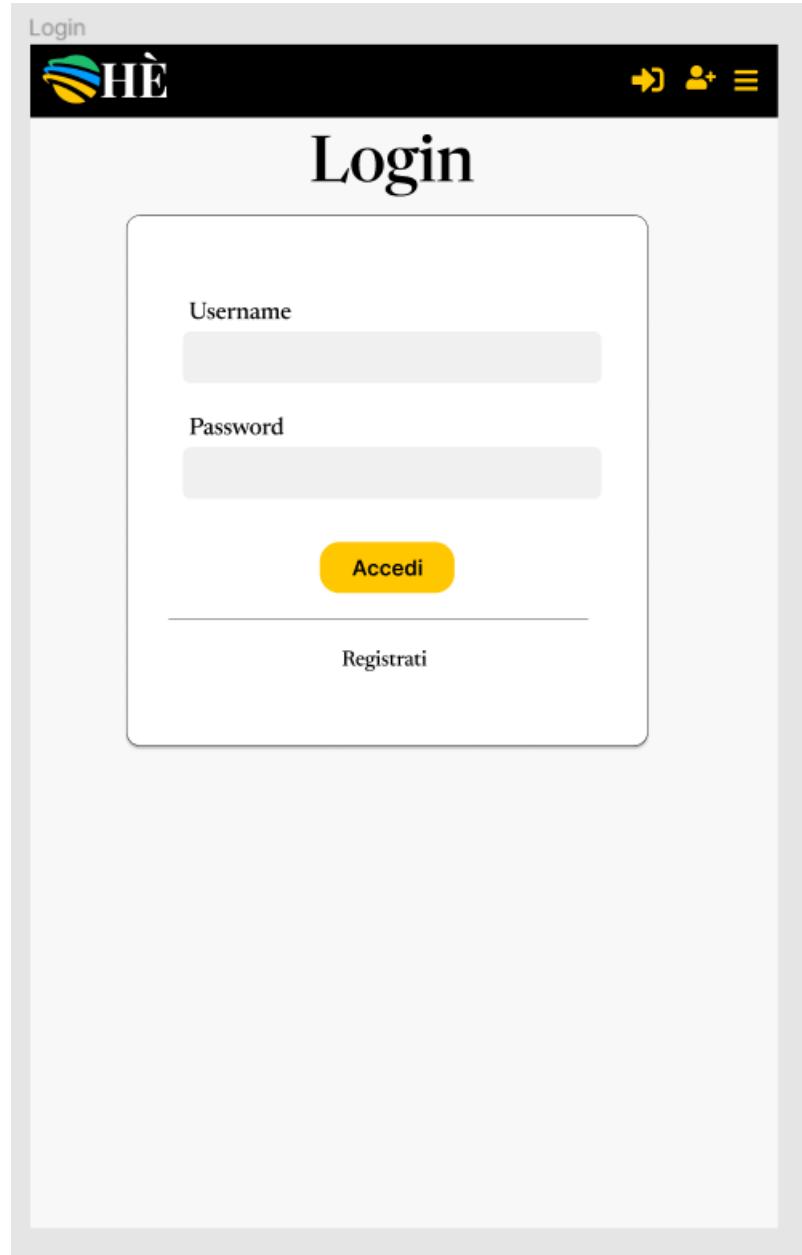


Figure 4: Mockup schermata di login

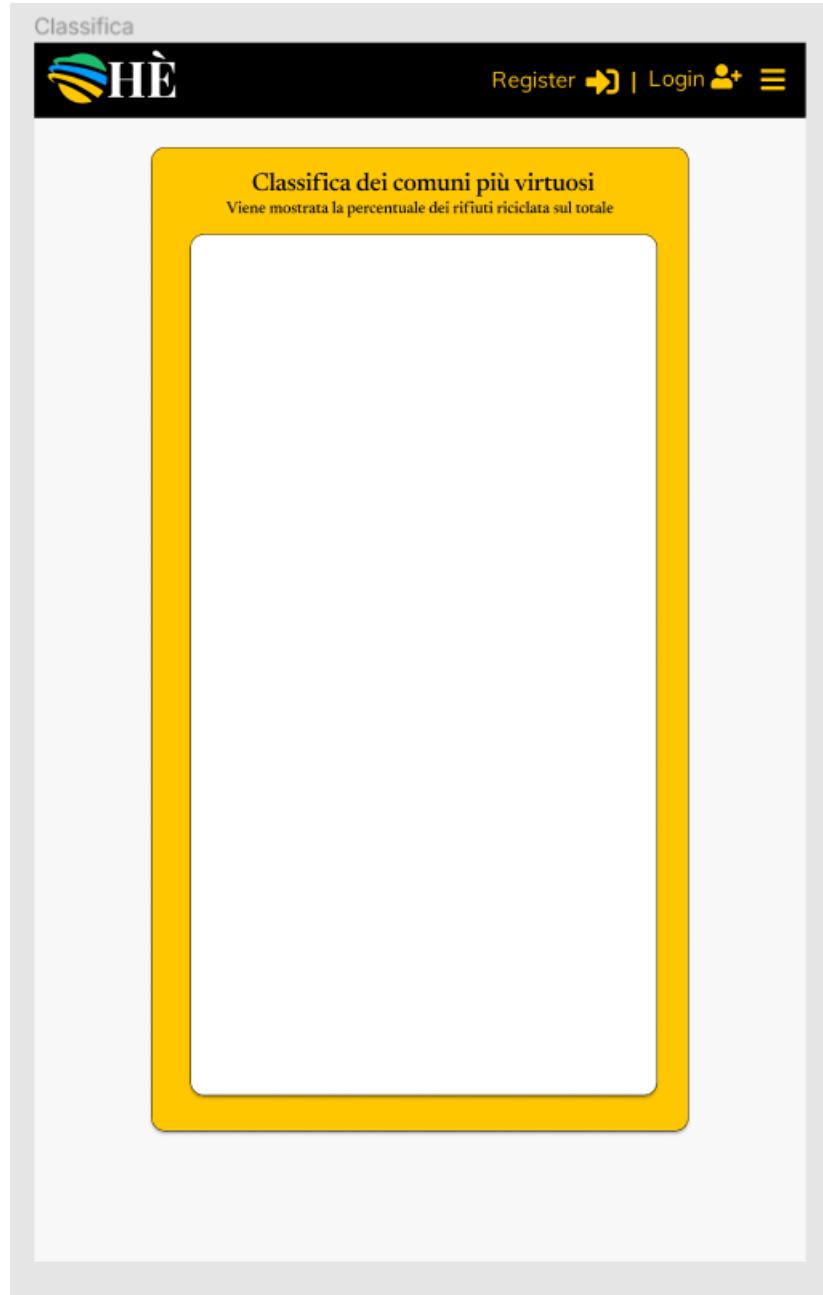


Figure 5: Mockup della pagina della classifica

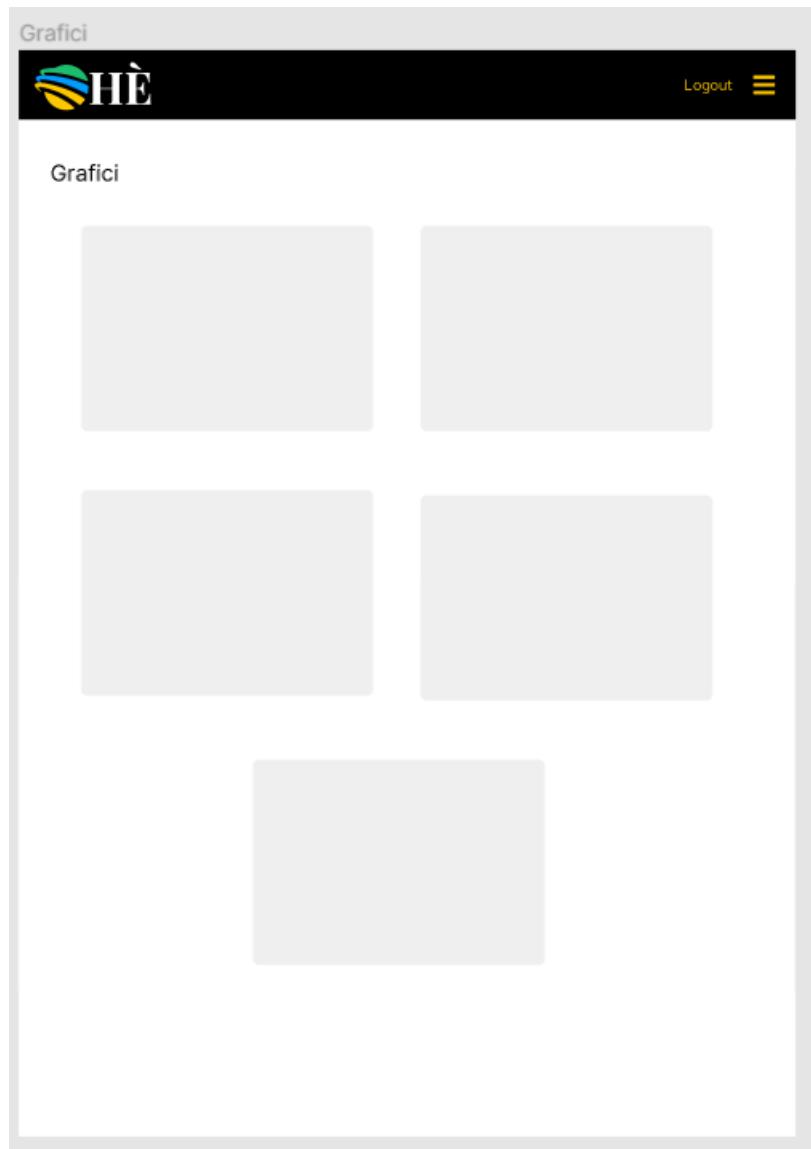


Figure 6: Mockup della pagina dei grafici

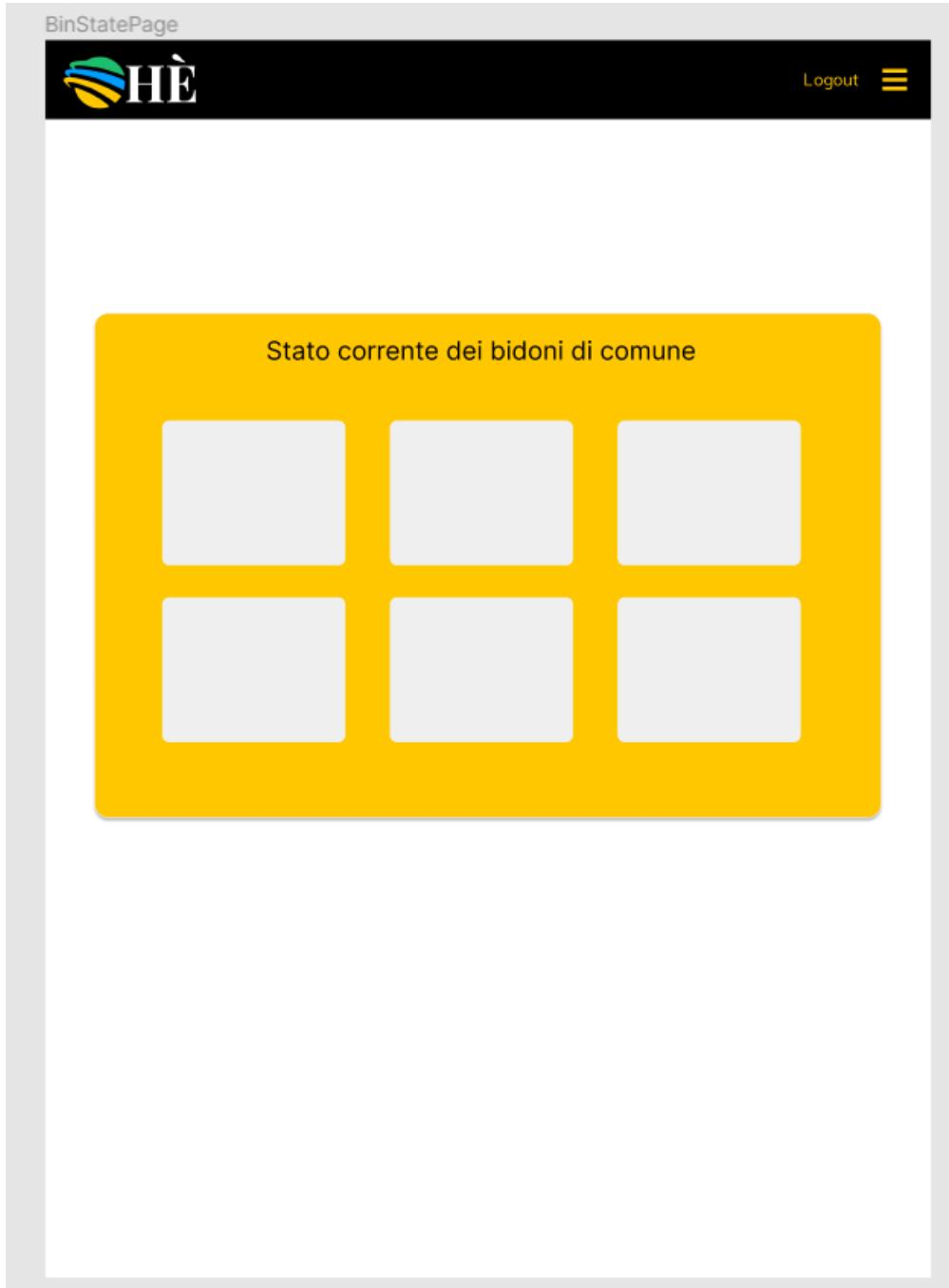


Figure 7: Mockup della pagina di visualizzazione dello stato dei bidoni

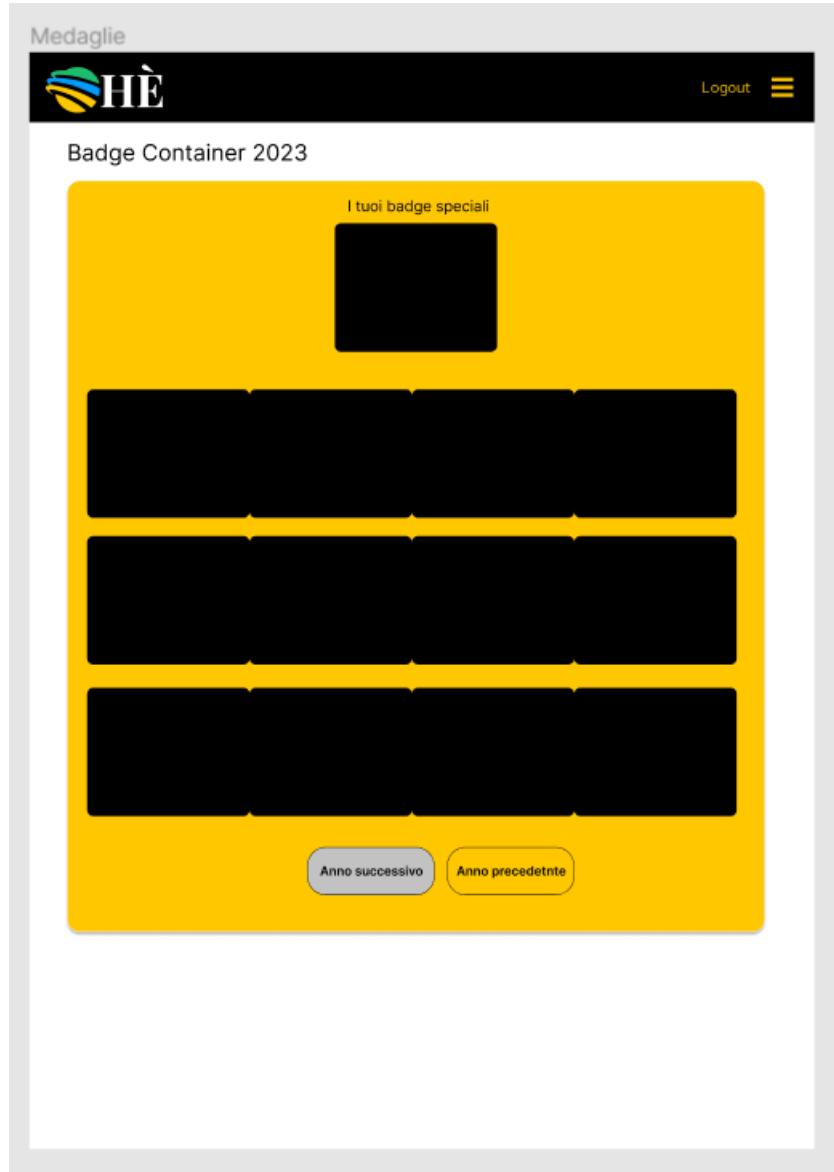


Figure 8: Mockup della pagina delle medaglie

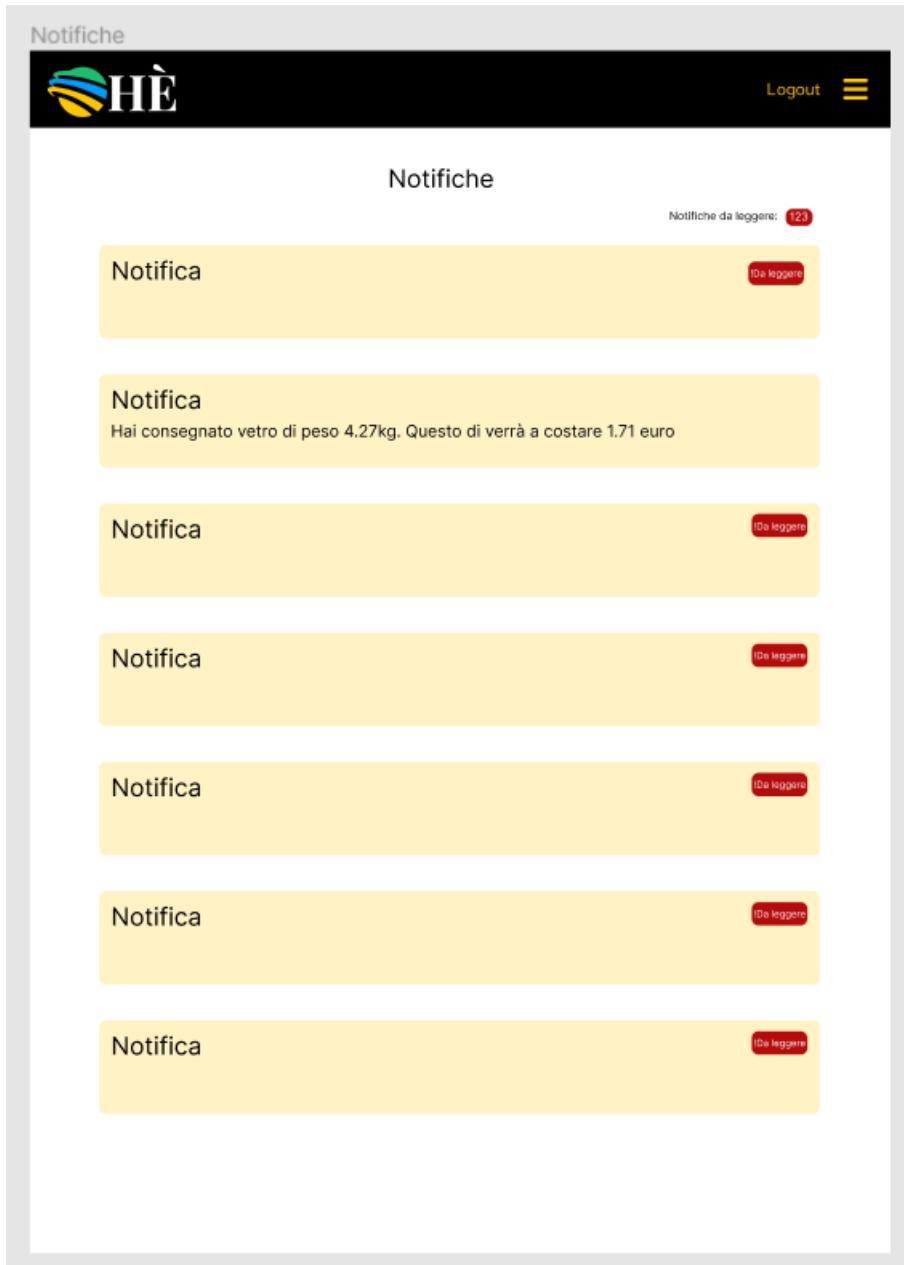


Figure 9: Mockup della pagina delle notifiche

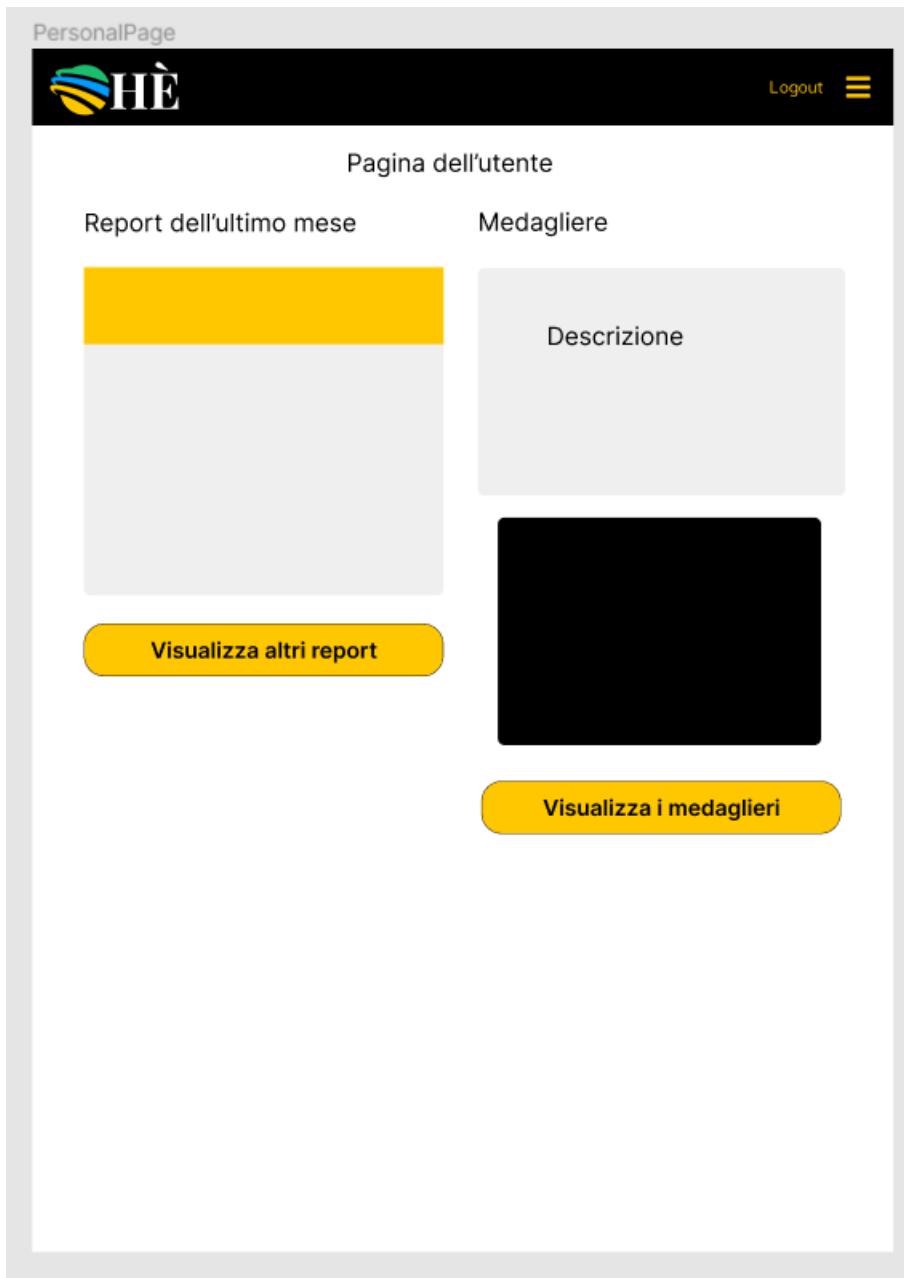


Figure 10: Mockup della pagina personale dell'utente

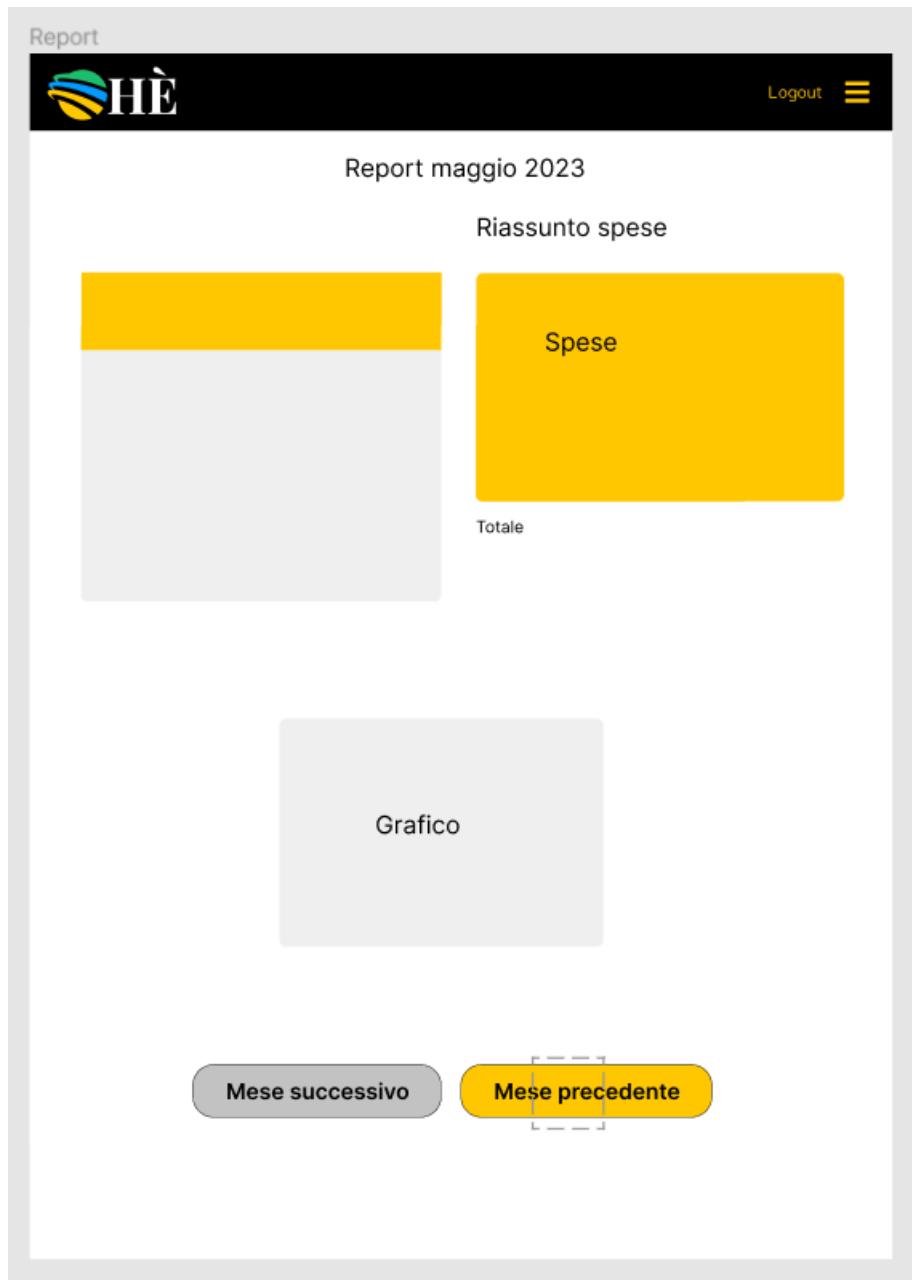


Figure 11: Mockup della pagina dei report

3.2 Analisi Target Users

Personas: Gianfranco Barbagianni

Gianfranco è un uomo di mezz'età preoccupato perché le bollette dei rifiuti stanno aumentando, e vorrebbe monitorare le sue spese in modo più chiaro.

- Gianfranco effettua l'accesso ogni volta che deposita dei rifiuti per controllare nelle notifiche quanto gli costerà il deposito
- Gianfranco accede quando arriva la bolletta dei rifiuti per controllare se i dati corrispondono, in particolare controlla spesso i report mensili per avere un paragone con i mesi precedenti

Personas: Gianpaolo Civetta

Gianpaolo è un giovane ragazzo ambientalista appassionato di statistiche, che vorrebbe avere sempre aver modo di conoscere i suoi consumi per poter migliorare le sue abitudini.

- Gianpaolo osserva molto spesso i grafici generali, per avere un'idea di come stanno andando le cose a livello globale senza perdere tempo effettuando il login
- Gianpaolo effettua spesso il login per monitorare le sue statistiche e darsi degli obiettivi mensili

Personas: Pierpaolo Gufo

Pierpaolo è un uomo afflitto da un problema: Essendo molto impegnato deposita raramente molti rifiuti in una volta sola, ma i bidoni del suo comune sono spesso pieni, e deve sempre usare la macchina per depositare i rifiuti in modo da poter cercare più velocemente un bidone vuoto, o per depositare i rifiuti in più bidoni diversi.

- Da quando ha scoperto il sito di Hè Pierpaolo vi accede ogni volta che deve depositare la spazzatura per sapere quale cassetto è libero in modo da non dover fare più viaggi

Personas: Piergiorgia Rapaci

Piergiorgia abita in un piccolo comune ed è madre di 2 figli, le sue bollette dei rifiuti sono sempre molto alte perché i figli buttano tutto nell'indifferenziata

- Piergiorgia effettua l'accesso al sito ogni ultimo del mese perchè ai figli piacciono molto le medaglie che si possono ottenere. In questo modo li ha convinti a utilizzare i bidoni in modo appropriato

- Piergiorgia visita il sito senza fare login anche per visualizzare con i figli la classifica dei comuni, infatti i figli sono molto competitivi e gli piace vedere il loro comune che scala la classifica per vantarsi con i compagni di scherma degli altri comuni

3.3 Storyboard

All'apertura il sito si apre sulla pagina Home che appare come segue (Figura 12).

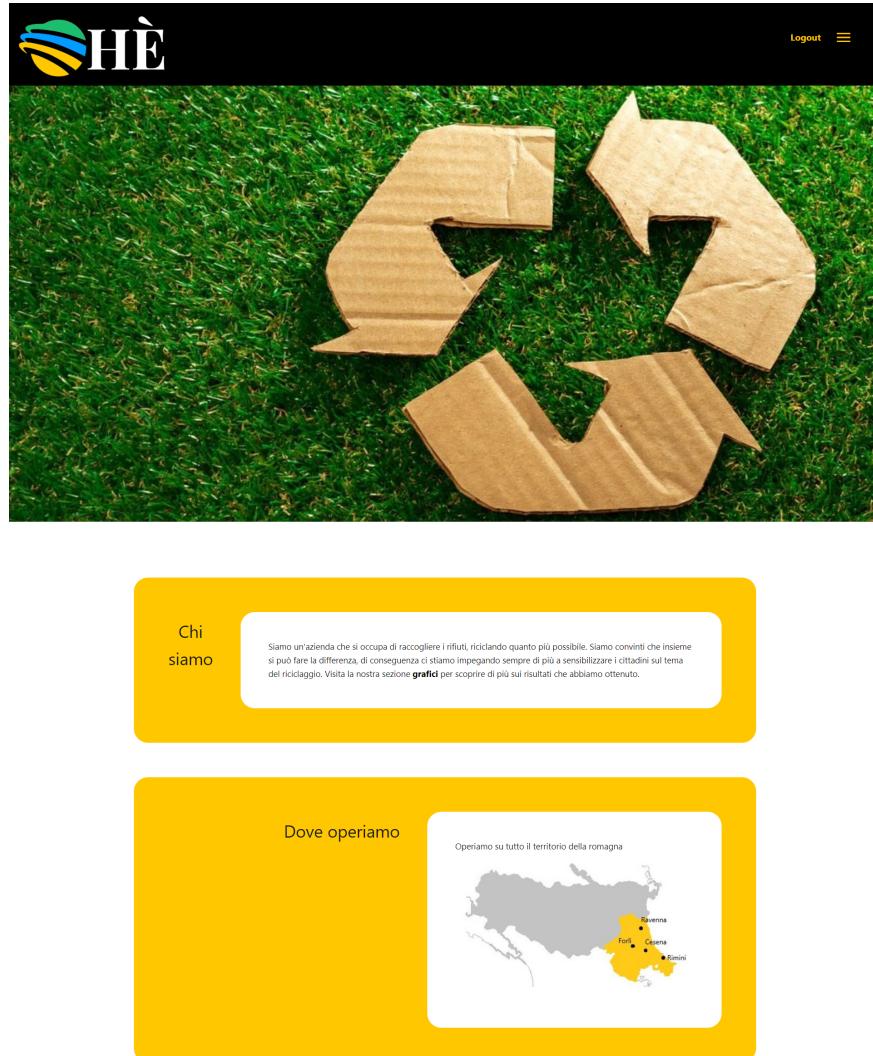


Figure 12: Home Page

Dalla pagina Home, tramite l'uso del menù laterale, l'utente potrà visualizzare la pagina contenente i grafici sulle statistiche raccolte dal sito su tutti i suoi utenti in merito alla gestione dei rifiuti e al riciclaggio (Figura 13).

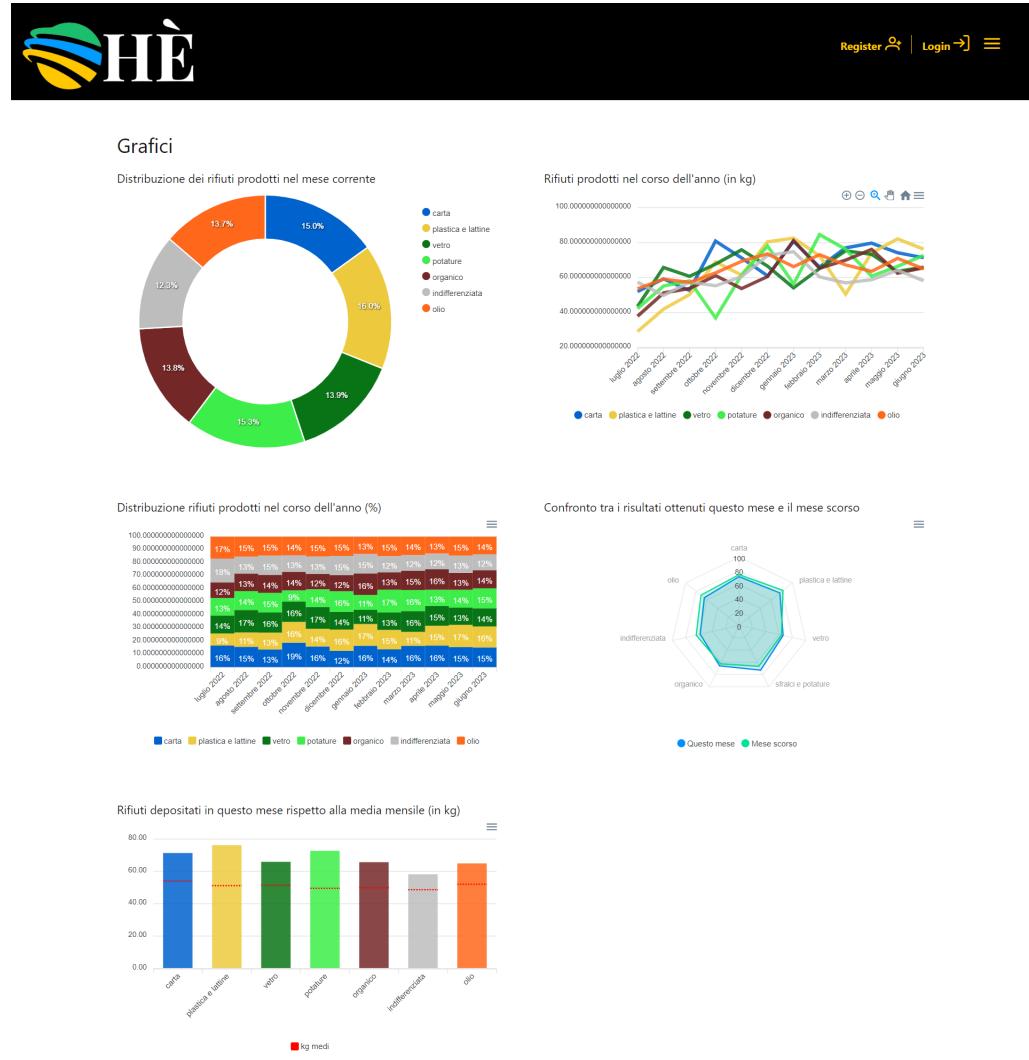


Figure 13: Graph Page

L'utente potrà anche visualizzare la classifica dei comuni più virtuosi in ambito del riciclaggio come visibile in Figura 14.

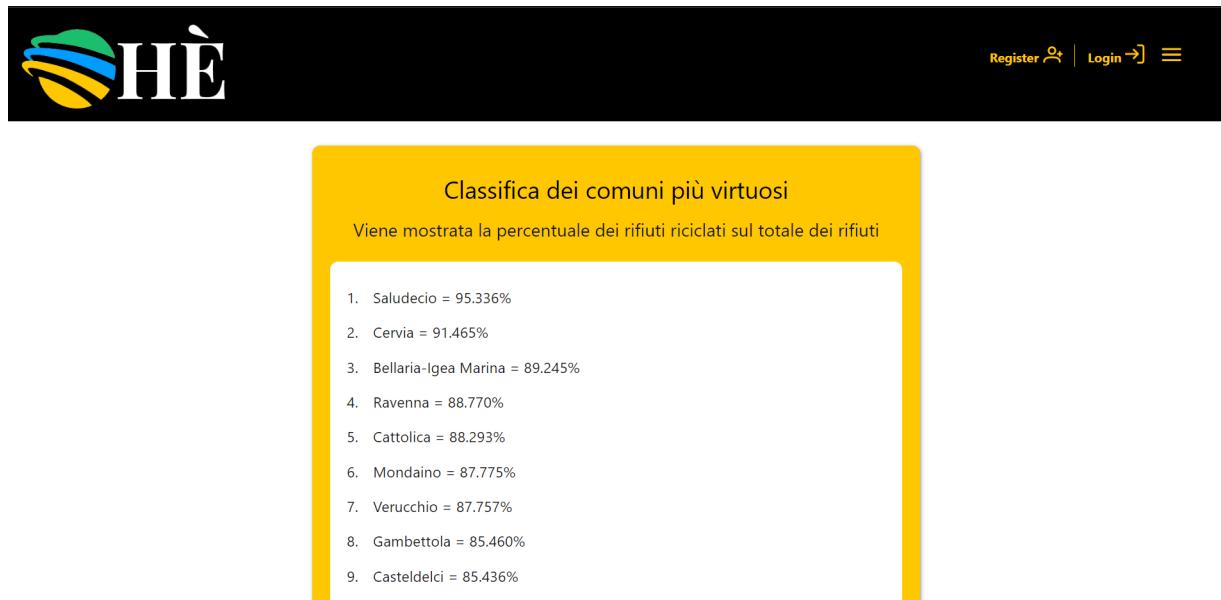


Figure 14: Classification Page

L'utente potrà registrarsi all'applicazione tramite l'apposita schermata (Figura 15).

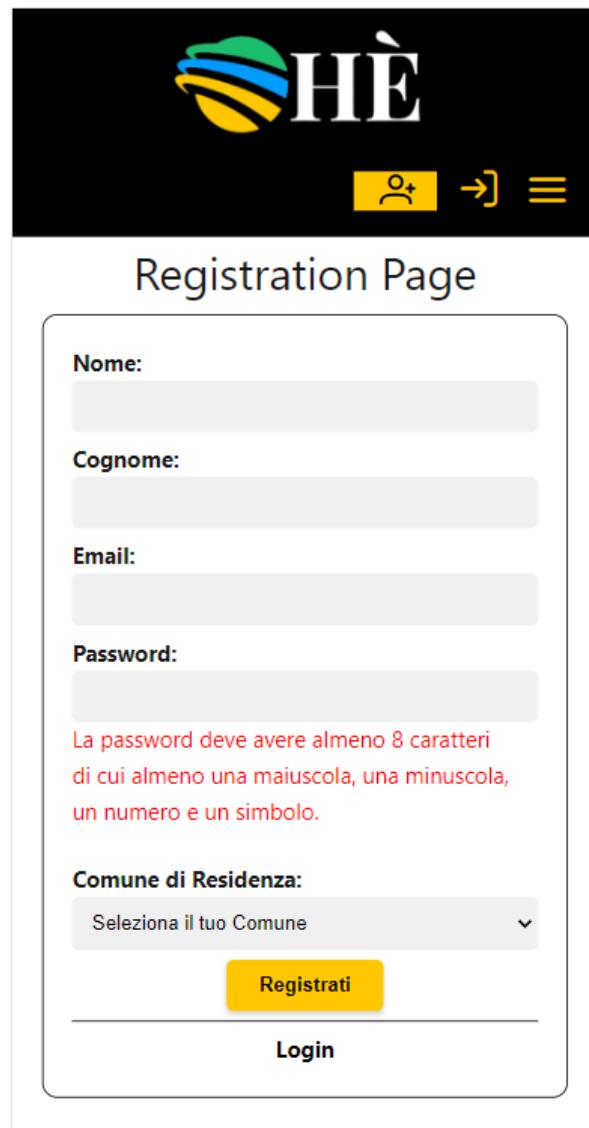


Figure 15: Register Page (mobile version)

Oppure nel caso si sia già registrato, tramite la schermata di login potrà autenticarsi direttamente nel sito (Figura 16).

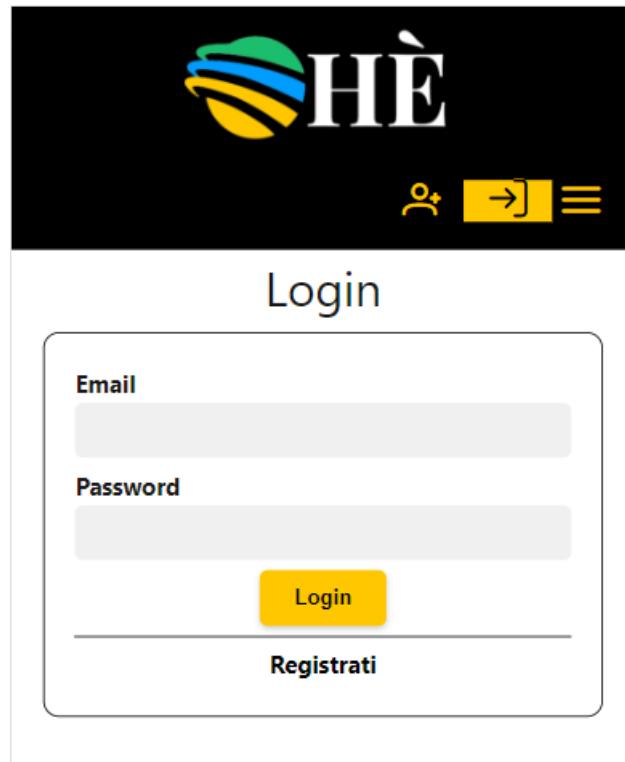


Figure 16: Login Page (mobile version)

Dopo essersi autenticati nel sito si verrà reindirizzati alla pagina personale (Figura 17).


Logout ≡

Pagina di Giuseppe Ricci

Report dell'ultimo mese

Categoria	maggio 2023	aprile 2023	Variazione
vetro	1.71 Kg	4.19 Kg	-59% ↓
plastica e lattine	6.48 Kg	2.01 Kg	222% ↑
carta	8.05 Kg	1.65 Kg	388% ↑
organico	4.57 Kg	1.69 Kg	170% ↑
sfralci e potature	3.27 Kg	1.42 Kg	130% ↑
olio	1.48 Kg	3.18 Kg	-53% ↓
indifferenziata	4.85 Kg	2 Kg	142% ↑

Medagliere

Queste sono le tue medaglie speciali! Forza! Collezionale tutte!

Cosa sono

Ogni mese vengono assegnate delle medaglie agli utenti che si impegnano particolarmente nel riciclaggio. Esiste una medaglia per ogni tipo di rifiuto, e se in un mese riesci ad ottenerle tutte ti guadagnerai una medaglia extra!

Come si ottengono

Puoi ottenere medaglie riciclando più dell'utente medio o migliorando rispetto al mese scorso, se ottieni un numero di medaglie pari a 5, 20, 50, 75 o 100 per una tipologia di rifiuto otterrai una medaglia speciale per quella tipologia!



[Visualizza altri Report](#)

[Visualizza i medaglieri](#)

Figure 17: Personal Page

Dalla pagina personale l'utente potrà facilmente raggiungere la pagina contenente i report mensili (Figura 18).


Logout ≡

Report maggio 2023

Categoria	maggio 2023	aprile 2023	Variazione
vetro	1.71 Kg	4.19 Kg	-59% ↓
plastica e lattine	6.48 Kg	2.01 Kg	222% ↑
carta	8.05 Kg	1.65 Kg	388% ↑
organico	4.57 Kg	1.69 Kg	170% ↑
sfralci e potature	3.27 Kg	1.42 Kg	130% ↑
olio	1.48 Kg	3.18 Kg	-53% ↓
indifferenziata	4.85 Kg	2 Kg	142% ↑

Riassunto spese

- Hai prodotto 1.71Kg di vetro al prezzo di 0.4 al Kg
- Hai prodotto 6.48Kg di plastica e lattine al prezzo di 0.9 al Kg
- Hai prodotto 8.05Kg di carta al prezzo di 0.3 al Kg
- Hai prodotto 4.57Kg di organico al prezzo di 0.4 al Kg
- Hai prodotto 3.27Kg di sfralci e potature al prezzo di 0.15 al Kg
- Hai prodotto 1.48Kg di olio al prezzo di 1 al Kg
- Hai prodotto 4.85Kg di indifferenziata al prezzo di 1.5 al Kg

Per un totale di 20€

Confronto ultimi 2 mesi



● This Month ● Last month

mese successivo

mese precedente

Figure 18: Report Page

Oppure potrà accedere alla pagina contenente i medaglieri (Figura 19).

The screenshot shows a yellow-themed dashboard titled "Badge Container 2023". At the top, there's a logo consisting of three overlapping semi-circles in green, blue, and yellow, followed by the letters "HÈ". On the right side, there are "Logout" and menu icons. Below the title, a section titled "I tuoi badge speciali" displays eight achievement icons arranged in two rows of four. The main area contains a grid of badge tracks for each month from January 2022 to May 2023. Each track consists of four horizontal rows of icons representing different actions or achievements. The tracks are labeled with months: maggio 2023, aprile 2023, marzo 2023, febbraio 2023, gennaio 2023, dicembre 2022, novembre 2022, ottobre 2022, settembre 2022, agosto 2022, luglio 2022, and giugno 2022. At the bottom of the page are two buttons: "anno successivo" (next year) and "anno precedente" (previous year).

Figure 19: Badge Page

Sempre utilizzando il menù laterale si potrà accedere alla pagina relativa allo stato dei bidoni del proprio comune (Figura 20).

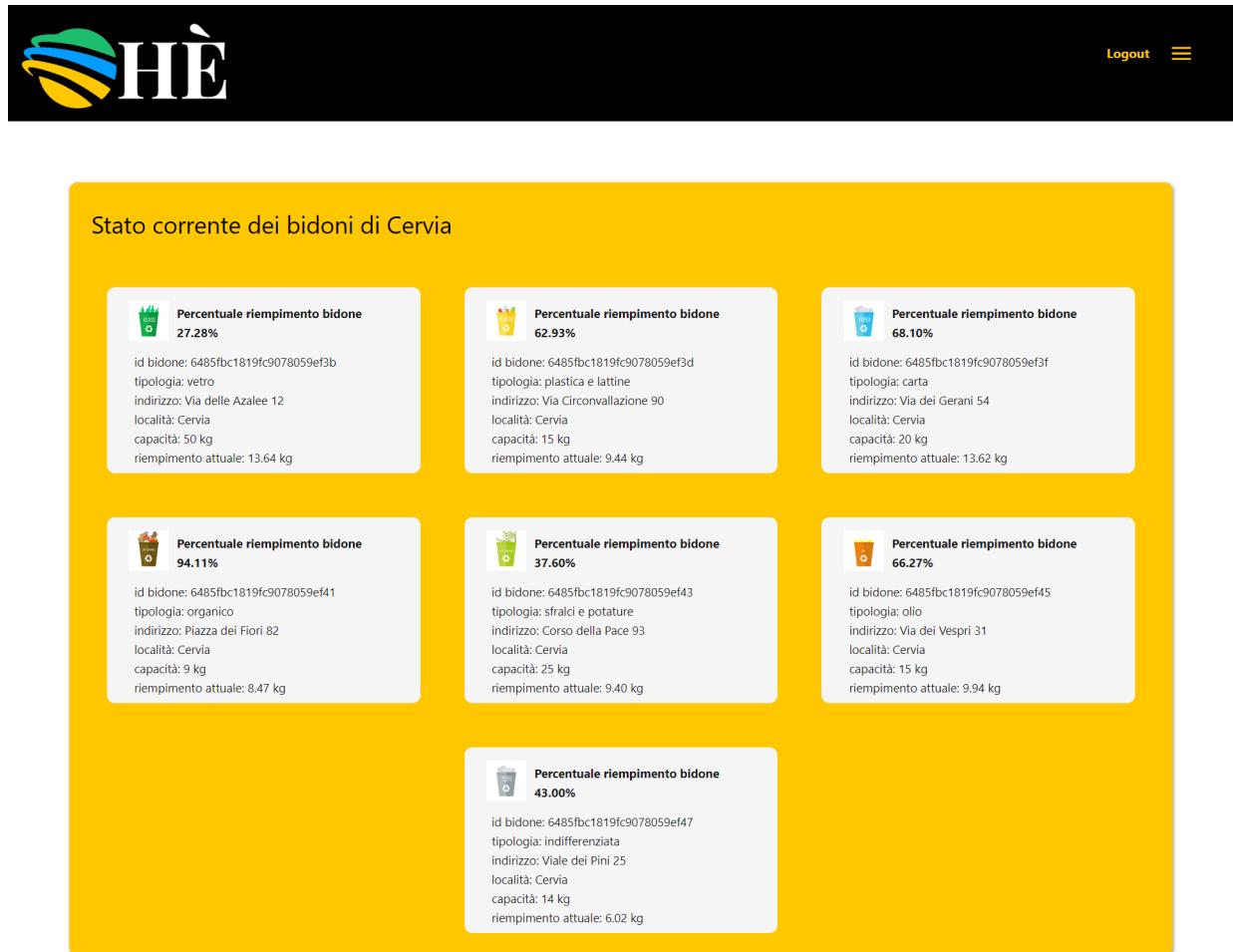


Figure 20: Bin State Page

E si potrà anche accedere alla pagina relativa ai grafici personali (Figura 21).



Figure 21: User Graph Page

Infine si potrà accedere alla pagina delle notifiche (Figura 22).

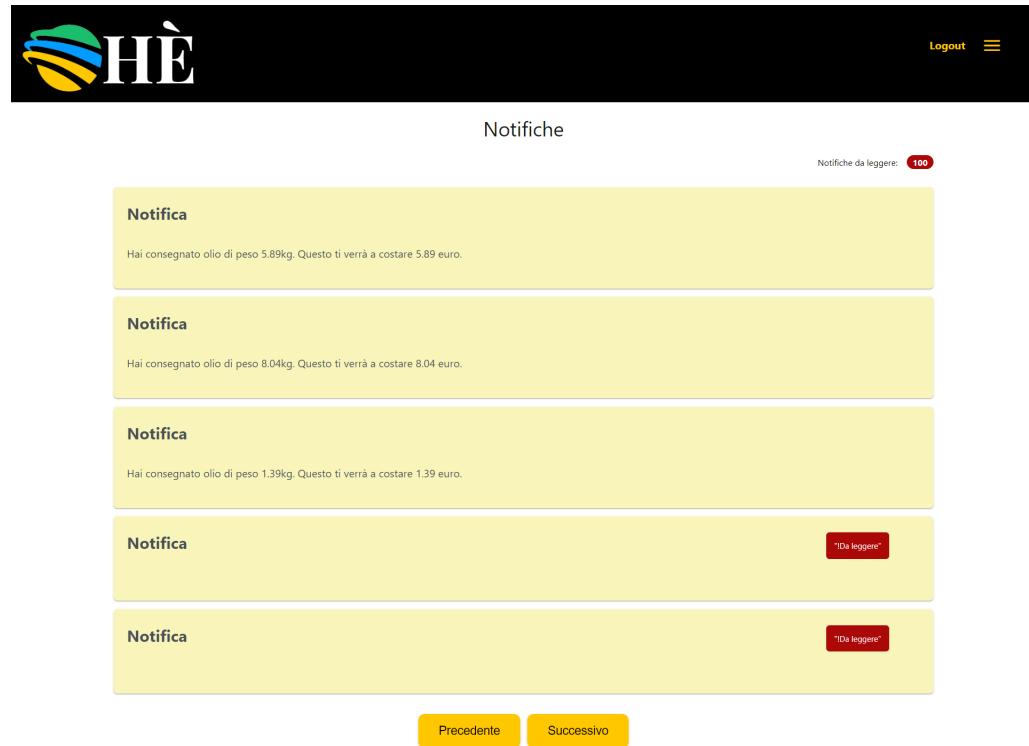


Figure 22: Notification Page

4 Tecnologie

Il sito è basato su un'architettura MEVN (piattaforma JavaScript fullstack per applicazioni web):

- **MongoDB [6]** : gestore di database NoSQL.
- **Express.js**: framework di sviluppo di Javascript lato server.
- **Vue.js [13]** : framework di sviluppo di Javascript lato client.
- **Node.js**: ambiente di esecuzione per eseguire applicazioni server-side senza l'utilizzo del browser.

Di seguito si elencano le tecnologie usate:

- **Nodemon**: per sviluppare applicazioni node.js riavviando automaticamente l'esecuzione ogni qualvolta i file sorgenti vengono modificati
- **Socket.io [10]** : È una libreria per applicazioni web real-time con comunicazioni bidirezionali tra client e server. Le comunicazioni avvengono inviando e ricevendo messaggi con le websocket. Socket.io è stata utilizzata per l'invio delle notifiche e per l'aggiornamento dei dati dei grafici.
- **Axios [3]** : È una libreria per effettuare richieste HTTP asincrone che si basa sul concetto di promise. In questo modo si garantisce che, non appena sarà possibile, la richiesta verrà esaudita.
- **Jsonwebtoken [5]** : Sono utilizzati nei sistemi di autenticazione per fornire un maggiore livello di sicurezza. Il server presente nel backend fornisce delle API Rest tramite le quali vengono trasferiti e ricevuti i Json Web Token.
- **Bcrypt**: Per la funzione di hashing delle password
- **Mongoose**: Per l'interfacciamento al database MongoDB
- **Apexcharts [2]** : È una libreria moderna per la creazione di grafici interattivi
- **Primevue [8]** : È una libreria per le componenti Vue

Linguaggi utilizzati:

- **HTML**
- **CSS**
- **SCSS**
- **TypeScript [11]**
- **Javascript**

4.1 frontend

L'interfaccia utente è stata realizzata tramite Vue ed è strutturata in pagine ognuna realizzata tramite components.

- **HomePage:** questa pagina contiene un'introduzione all'applicazione che descrive il ruolo che essa ricopre e la zona su cui opera.
- **GraphPage:** questa pagina contiene le varie statistiche mostrate dalla pagina in formato di grafici, è composta da una serie di components come PieGraph e RadarGraph che rappresentano la specifica tipologia di grafico mostrato e da un GraphContainer che permette di organizzare i titoli e la posizione dei grafici al suo interno.
- **ClassificationPage:** questa pagina mostra la classifica dei comuni in base alla percentuale di rifiuti riciclati rispetto ai rifiuti totali prodotti dagli utenti di ogni comune.
- **RegisterPage e LoginPage:** queste due pagine servono rispettivamente per registrarsi e autenticarsi all'applicazione; entrambe sono costituite da due components: un container e un form. Dopo aver eseguito correttamente il login si viene reindirizzati alla PersonalPage.
- **PersonalPage:** questa pagina svolga la funzione di home per l'area riservata all'utente corrente e contiene alcuni degli elementi principali delle altre pagine riservate riutilizzando components come ReportTable e SpecialBadgeContainer e aggiungendo un link alle pagine specifiche in modo da invogliare l'utente a visitarle.
- **ReportPage:** questa pagina è dedicata a offrire informazioni sui quantitativi di rifiuti riciclati per ogni tipologia(vetro, plastica...) e a confrontare gli ultimi due mesi trascorsi per fornire un'idea all'utente dei suoi eventuali cambiamenti a livello di produzione di rifiuti. La pagina fornisce anche un semplice grafico sempre per comparare gli ultimi due mesi e un riassunto dei pagamenti dovuti sulla base dei rifiuti prodotti.
- **BinStatePage:** l'obiettivo di questa pagina è quello di offrire agli utenti un quadro dello stato attuale di riempimento dei bidoni nel loro comune. In questo modo sarà possibile visualizzare se un bidone è attualmente pieno o vicino al riempimento e in caso utilizzarne un altro con uno stato di riempimento inferiore.
- **NotificationPage:** questa pagina raccoglie tutte le notifiche prodotte dal sistema ogni volta che effettui un deposito riportando il tipo di rifiuti, il quantitativo, e il costo calcolato dal tipo e dal peso. Vengono inoltre mostrate le notifiche che ogni mese ti avvisano del report del mese precedente.

- **BadgePage:** questa pagina mostra i premi ottenuti dall'utente sotto forma di medagliere; l'utente riceve medaglie sulla base del quantitativo di rifiuti riciclabili prodotti. Sono anche presenti delle medaglie speciali che si completano ottenendo numerose medaglie in modo da incentivare un impegno continuativo.
- **UserGraphPage:** questa pagina, molto simile a GraphPage, mostra i grafici relativi al singolo utente riutilizzando gli stessi components ma utilizzando un container differente per le differenze nei dati utilizzati.

4.2 schema db

Il database Mongo contiene svariati schemi, ovvero:

- **Municipalities:** schema che rappresenta i comuni. È composto da:
 - **name:** nome del comune
 - **users:** numero degli utenti di tale comune
- **Users:** schema che rappresenta gli utenti. È composto da:
 - **name:** nome dell'utente
 - **surname:** cognome dell'utente
 - **email:** email dell'utente
 - **password:** password dell'utente crittografata
 - **salt:** sale della password
 - **municipality:** id del comune dell'utente
 - **date:** data di registrazione, composta dai campi year e month
- **Typology:** schema che rappresenta le tipologie di rifiuto e le informazioni ad esso correlate. È composto da:
 - **name:** nome della tipologia di rifiuto
 - **max_kg:** capienza massima dei bidoni per quel tipo di rifiuto (in kg)
 - **price_kg:** prezzo al kg per la tipologia di rifiuto
- **Badges:** schema che rappresenta una tipologia di medaglia da assegnare all'utente. È composto da:
 - **name:** nome della tipologia di medaglia
 - **is_multiple:** booleano, indica se la medaglia è speciale, ovvero della tipologia assegnata quando sono presenti un certo numero di medaglie normali con lo stesso nome
 - **repetition:** campo opzionale, presente solo se IS_MULTIPLE è TRUE, rappresenta numero di medaglie normali necessarie per ottenere la medaglia speciale

- **Bins:** schema che rappresenta uno specifico bidone. È composto da:
 - **typology:** id della tipologia di rifiuto
 - **actual_kg:** kg di rifiuti presenti nel bidone
 - **address:** indirizzo in cui si trova bidone
 - **municipality:** id del comune in cui si trova il bidone
- **Deposit:** schema che rappresenta un deposito di rifiuti. È composto da:
 - **user:** id dell'utente che ha fatto il deposito
 - **kg:** kg depositati
 - **bin:** id del bidone in cui è stato fatto il deposito
 - **createdAt:** data di creazione del deposito
- **Notifications:** schema che rappresenta le notifiche ricevute dagli utenti. È composto da:
 - **email:** email dell'utente
 - **type:** tipo della notifica, assume valori deposit o report
 - **isRead:** indica se la notifica è stata letta o meno
 - **text:** testo della notifica
 - **date:** data di creazione della notifica, composta dai campi year e month
- **Reports:** schema che rappresenta i report mensili con il confronto rispetto al mese precedente. È composto da:
 - **user:** id dell'utente riguardo cui è il report
 - **quantities:** array composto da nome della tipologia di rifiuto, kg del mese corrente e del mese precedente
 - **date:** data di creazione report, composta dai campi year e month
- **UserBadges:** schema che rappresenta le medaglie ottenute dai singoli utenti. È composto da:
 - **user:** id dell'utente che ha ottenuto la medaglia
 - **badge:** id della tipologia di medaglia ottenuta
 - **createdAt:** data di creazione della medaglia

4.3 backend

Il backend offre svariate API per ottenere dati da mongodb. Tali API sono sviluppate tramite axios e sono:

- **/badge (get)** questa API restituisce un'array contenente tutti i badge ottenuti da uno specifico utente a partire da una data, passatagli tramite params, fino alla stessa data dell'anno precedente
- **/binState (get)** restituisce lo stato dei bidoni del comune dell'utente attuale
- **/classification (get)** restituisce la classifica dei comuni più virtuosi calcolata in base alla percentuale di rifiuti riciclati rispetto ai totali
- **/login (post)** permette all'utente di effettuare il login
- **/municipality/names (get)** restituisce tutti i comuni
- **/register (post)** permette la registrazione di un nuovo utente salvando i suoi dati su mongoDB
- **/report (get)** restituisce i dati relativi al report calcolati sulla base del mese passatogli tramite props. Se il report non è presente sul database viene creato e caricato prima di essere restituito. Inoltre in questa fase se viene creato un report vengono creati e inseriti nel database i badge calcolati sulla base del report creato.
- **/typology/price (get)** restituisce il prezzo al kg delle varie tipologie di rifiuto
- **/user/verify (get)** verifica se l'utente corrente è autenticato o meno
- **/user/registrationDate (get)** restituisce la data della registrazione dell'utente (mese, anno)
- **/user/fullname (get)** restituisce nome e cognome dell'utente corrente
- **/deposit (post)** aggiunge un deposito da parte di un utente in uno specifico bidone

Inoltre è stato utilizzato socket.io per le query mongo che richiedevano un aggiornamento frequente, ovvero per le notifiche e i grafici.

4.4 sicurezza

Per garantire la sicurezza degli account degli utenti, ogni volta che viene effettuata una registrazione, la password non viene salvata in chiaro ma viene sia salata che pepata. Inoltre per evitare accessi non autorizzati dovuti alla disattenzione dell'utente ogni sessione ha la durata massima di 1 ora dopo la quale sarà necessario effettuare nuovamente il login.

4.5 Gamification

All'interno dell'applicazione sono stati inseriti degli elementi di gamification volti a intrattenere l'utente mentre utilizza l'applicazione e allo stesso tempo promuovere comportamenti positivi in materia di riciclaggio. Ogni mese l'utente riceve delle medaglie attribuite sulla base di quanto l'utente sia migliorato nel riciclaggio come la produzione un quantitativo di rifiuti riciclati superiore rispetto ai rifiuti indifferenziati oppure, più semplicemente, hai riciclato di più dell'utente medio un determinato tipo di rifiuto. Ottenendo queste medaglie ogni mese si possono ottenere alcune medaglie speciali a rappresentare l'impegno continuativo. Infatti queste ultime cambiano aspetto sulla base di quante medaglie normali si sono ottenute andandosi a completare solo dopo un lungo periodo.

5 Codice

Solo aspetti rilevanti.

5.1 interazione con Vuex

Per poter gestire le pagine dei report e delle medaglie, dato che cambiano rispettivamente per mese e per anno, abbiamo deciso di utilizzare Vuex per gestire 2 coppie di variabili per poter gestire il cambiamento di mese/anno. Viene riportato lo store Vuex utilizzato per mantenere e cambiare queste variabili.

```
import { createStore, Commit, Store } from 'vuex'

interface RootState {
  reportMonth: number
  reportYear: number
  badgeMonth: number
  badgeYear: number
}

const store = createStore<RootState>({
  state: {
    reportMonth: new Date().getMonth(),
    reportYear: new Date().getFullYear(),
    badgeMonth: new Date().getMonth(),
    badgeYear: new Date().getFullYear(),
  },
  mutations: {
    updateReportMonth(state: RootState, month: number) {
      state.reportMonth = month
    },
  },
})
```

```

updateReportYear(state: RootState, year: number) {
    state.reportYear = year
},
updateBadgeMonth(state: RootState, month: number) {
    state.badgeMonth = month
},
updateBadgeYear(state: RootState, year: number) {
    state.badgeYear = year
},
},
actions: {
    setReportMonth({ commit }: { commit: Commit },
        month: number) {
        commit('updateReportMonth', month)
    },
    setReportYear({ commit }: { commit: Commit },
        year: number) {
        commit('updateReportYear', year)
    },
    setBadgeMonth({ commit }: { commit: Commit },
        month: number) {
        commit('updateBadgeMonth', month)
    },
    setBadgeYear({ commit }: { commit: Commit },
        year: number) {
        commit('updateBadgeYear', year)
    },
},
getters: {
    getReportMonth: (state: RootState) =>
        state.reportMonth,
    getReportYear: (state: RootState) =>
        state.reportYear,
    getBadgeMonth: (state: RootState) =>
        state.badgeMonth,
    getBadgeYear: (state: RootState) =>
        state.badgeYear,
},
})
export default store

```

5.2 Gestione per la simulazione dei depositi

Per la gestione dei depositi è possibile effettuare una richiesta POST al server per crearne uno fornendo la mail di un utente, un quantitativo e l'id di un bidone. Prima di aggiungere il deposito al database sarà verificato che il bidone non sia già pieno e, nel caso di un deposito con successo, verrà creata la notifica da inviare all'utente relativa al deposito.

```
const handleDeposit = async (req, res) => {
  const { email, kg, binId } = req.body;
  try {
    //Obtain user
    const user = await User.findOne({ email });
    // Fetch the bin and typology details based on
    //the provided binId
    const bin = await Bin.findById(binId);
    if (bin == null) {
      return res.status(404).json(
        { error: 'Bin not found' });
    }
    const typology = await Typology
      .findById(bin.typology);
    if (typology == null) {
      return res.status(404).json(
        { error: 'Typology not found' });
    }
    const {actual_kg} = bin;
    // Calculate the total weight after the deposit
    const totalWeight = kg + actual_kg;
    // Check if the total weight exceeds the
    // typology's maximum weight limit
    if (totalWeight > typology.max_kg) {
      return res.status(400).json(
        { error: 'Weight limit exceeded' });
    }
    // Create the deposit record
    const deposit = new Deposit({
      user: user._id,
      kg,
      bin: binId,
    });
    // Save the deposit to the database
    await deposit.save();
    // Update the actual_kg in the bin
```

```

bin.actual_kg = totalWeight;
await bin.save();
// Creating Notification
const depositPrice = Math.round(
    (kg * typology.price_kg) * 100) / 100;
const currentDate = new Date();
const month = currentDate.getMonth() + 1;
const year = currentDate.getFullYear();
const notification = new Notification({
    email,
    date: {
        month,
        year
    },
    type: "deposit",
    isRead: false,
    text: "Hai consegnato " + typology.name +
        " di peso " + kg + "kg in " +
        bin.address + ". Questo ti verr " +
        " a costare " + depositPrice + " euro."
})
await notification.save()
return res.status(201).json(deposit);
} catch (error) {
    console.error(error);
    return res.status(500).json(
        { error: 'Internal server error' });
}
};

```

5.3 Richieste tramite Socket.io

Socket.io è una libreria che abilita la comunicazione bidirezionale tra client e server, per questo motivo avevamo pensato di utilizzare socket.io in maniera real-time con il server che inviava al client i dati ogni volta che essi venivano modificati ma, avendo riscontrato difficoltà nell'osservare i dati dei grafici in quanto cambiavano troppo velocemente, abbiamo optato per una versione in cui il client effettua una richiesta al server ogni minuto per ottenere dati aggiornati sui grafici. Viene riportato un esempio di richiesta sia lato client che server, in particolare quella per l'ottenimento dei dati del radar graph.

```

export function getRadarData(input: string) {
    return new Promise<number[][]>(
        (resolve, reject) => {socket.emit(
            'getRadarData', input,
            (response: number[][])) => {
                resolve(response);
            });
    });
}

```

```

socket.on('getRadarData', (data, callback) => {
    if(data=="general"){
        Radar.generalData().then((res)=>{
            callback(res);
        })
    }else{
        Radar.userData(data).then((res)=>{
            callback(res);
        })
    }
});

```

6 Test

L'applicazione è stata testata sui browser Chrome e Edge, sia in modalità desktop sia in modalità mobile, in modo da verificare la corretta visualizzazione dei vari elementi. Sono stati utilizzati **Contrast checker** [4] per la validazione dei colori, **AChecker**[1] per valutare l'accessibilità delle varie pagine e **W3C Validation**[12] per la validità del codice. Per il test delle API è stato utilizzato **Postman**[7].

6.1 Euristiche di Nielsen

- **Visibilità dello stato del sistema:** In alcune pagine che richiedono contenuti specifici, nel caso non siano disponibili, viene visualizzato un messaggio che indica il motivo della mancanza. Nelle pagine che richiedono di caricare una grossa mole di dati viene visualizzata una schermata di loading.
- **Corrispondenza tra sistema e mondo reale:** Le icone, i colori e la

simbologia utilizzata sono conformi a quelle comunemente utilizzate anche in relazione al dominio dell'applicazione.

- **Controllo e libertà:** L'utente è libero di muoversi liberamente tra le pagine del sito tramite un menù laterale. Nel caso l'utente non sia autenticato è comunque libero di esplorare le parti del sito in cui non è richiesta l'autenticazione e che riguardano informazioni disponibili a tutti.
- **Consistenza e standard:** Nell'intera applicazione sono ricorrenti gli stessi colori ed elementi in modo da ottenere un senso di uniformità. Inoltre, in tutte le pagine è presente il logo del sito.
- **Prevenzione dell'errore:** Qualora l'utente abbia eseguito l'accesso a un pagina differente da quella desiderata è sempre libero di cambiarla tramite il menù laterale. In ogni pagina i grafici e i vari elementi sono accompagnati da titoli e brevi spiegazioni per aiutare l'utente nella loro comprensione.
- **Design ed estetica minimalista:** In tutte le pagine viene indicato in modo chiaro e comprensibile all'utente il suo contenuto senza elementi estetici e di design che possano distrarre l'utente.
- **Documentazione:** Data la semplicità delle azioni eseguibili sul sito (Registrazione, Login, Navigazione tra pagine) non è necessario fornire una documentazione all'utente.

6.2 Test di usabilità

Ognuno dei membri ha testato durante l'intera durata del progetto le parti realizzate dagli altri membri del gruppo. I feedback forniti sono stati prevalentemente di natura grafica ed estetica e, più raramente, relativi alla chiarezza e alla comprensibilità del contenuto delle varie pagine.

7 Deployment

7.1 Installazione

Per l'installazione è necessario eseguire i seguenti step:

1. Clonazione repository [9]
2. Spostarsi all'interno della cartella del progetto utilizzando una shell Bash
3. eseguire i comandi:
 - (a) cd ./Hè/
 - (b) npm install
 - (c) cd ../backend/

(d) npm install

Proseguire con la messa in funzione.

7.2 Messa in funzione

Frontend:

Dall'interno della cartella Hè eseguire npm start

Backend:

Dall'interno della cartella backend eseguire npm start

È possibile collegarsi alla piattaforma digitando nel browser <http://127.0.0.1:5173/>

7.3 Simulazione

La prima volta che viene eseguito il backend il database viene prepopolato con dei dati fittizi.

Esempio di utente fittizio già presente:

1. email: francesca.rossi@example.com
2. email: giuseppe.ricci@example.com

Per ogni utente fittizio la password è Prova99#

8 Conclusioni

Il risultato ottenuto per questo progetto è un sistema che permette di rendere gli utenti più consci dell'impatto delle loro azioni quotidiane e che li stimoli a riciclare maggiormente per migliorare lo stato delle cose. Il sistema è espandibile in svariati modi, innanzitutto potrebbero essere inseriti più comuni per dare all'utente una visione della situazione nazionale, inoltre potrebbero essere aggiunti altri elementi di gamification, come dei premi per i comuni che sono stati più volte in cima alla classifica, visibili nella pagina degli utenti di tale comune, o dei grafici ulteriori con statistiche relative ai diversi comuni. Durante questo progetto ci siamo impegnati ad utilizzare svariate tecnologie diverse per osservarne i comportamenti e comprendere al meglio le interazioni tra esse.

Bibliography

- [1] AChecker. <https://achecker.achecks.ca/checker/index.php>.
- [2] Apexcharts. <https://apexcharts.com/>.
- [3] Axios. <https://axios-http.com/>.
- [4] Contrast Checker. <https://webaim.org/resources/contrastchecker/>.
- [5] JWT. <https://jwt.io/>.
- [6] MongoDB. <https://www.mongodb.com/it-it>.
- [7] Postman. <https://www.postman.com/>.
- [8] Primevue. <https://primevue.org/>.
- [9] Repository progetto. https://github.com/Sofy24/Progetto_ASW.git.
- [10] SocketIO. <https://socket.io/>.
- [11] Typescript. <https://www.typescriptlang.org/>.
- [12] W3C Validator. <https://validator.w3.org/>.
- [13] Vue. <https://vuejs.org/>.