Corso di Programmazione Web e Mobile

A.A. 2021-2022

TimeTimer

Gabriele Sammartino 954920

1. Introduzione

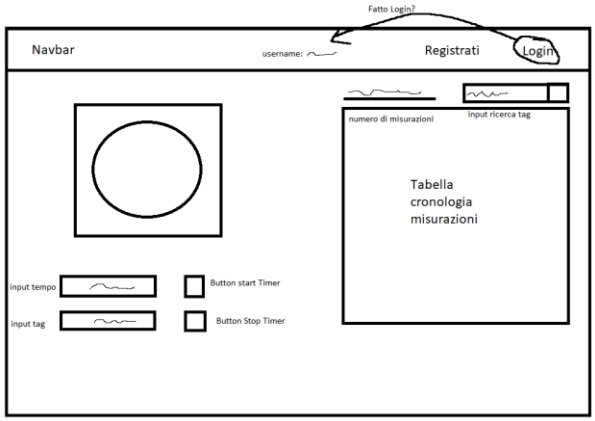
Il progetto è basato sul prodotto TimeTimer, un visual timer ideato per supportare la consapevolezza del trascorrere del tempo. Essendo bambini ed individui con disturbi dell'apprendimento il focus principale di TimeTimer, gli elementi del'app sono stati creati per migliorare l'usabilità da parte di utenti meno esperti nell'uso di dispositivi elettronici.

Una volta eseguito l'accesso all'app, gli utenti avranno la possibilità di inserire un valore (min: 1; max:60) equivalente al numero di minuti in cui il timer sarà in funzione. L'utente potrà inserire oltre al tempo, un tag, il quale permetterà di raggruppare le varie misurazioni in "sotto-gruppi". Ad ogni refresh della pagina, la tabella delle misurazioni verrà riempita con la cronologia di tutti i dati, organizzati come segue: tempo inserito dall'utente; tempo rimanente all'interruzione del timer; tag assegnato alla misurazione. Ad ogni misurazione viene anche posto un pulsante per cancellare tale dato. Inserendo un particolare tag nella barra di ricerca, verranno visualizzate nella tabella tutte e solo quelle misurazioni memorizzate aventi tale tag.

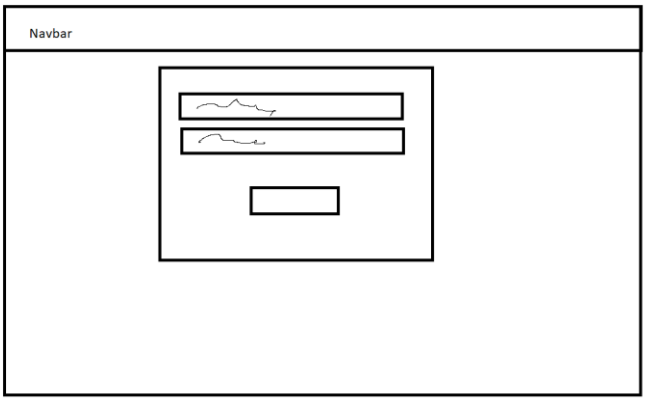
Le misurazioni e gli utenti vengono salvati in un DB realizzato utilizzando MongoDB. Ogni volta che viene registrato un'utente oppure un'utente vuole effettuare il login, viene interrogato il DB per controllare se l'utente è esistente dopo aver eseguito alcune funzioni di validità. Allo stesso tempo, il database viene interrogato anche quando un utente inserisce una misurazione o quando le misurazioni vengono prese per essere visualizzate a schermo.

Le interfacce dell'applicazione sono progettate con HTML e CSS, facendo uso di librerie esterne, quali jquery e bootstrap. Le richieste GET e POST vengono gestite attraverso chiamate AJAX verso il nostro server, implementato tramite Express, un framework di Node.js. Sarà il server ad interrogare il nostro DB, restituendo in caso di successo i dati da noi richiesti.

2. Interfacce

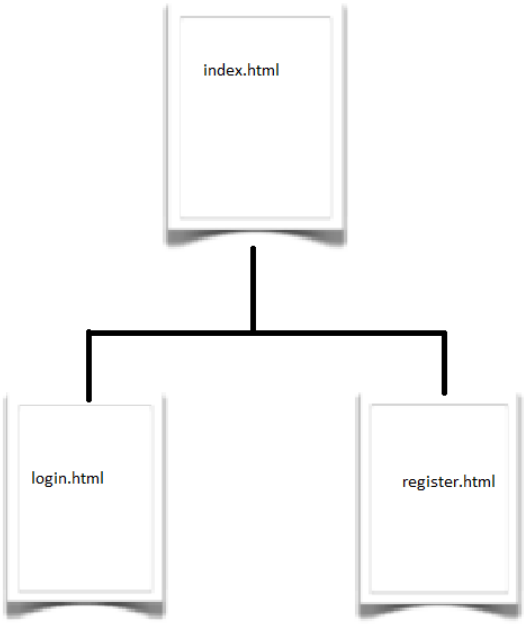


Index

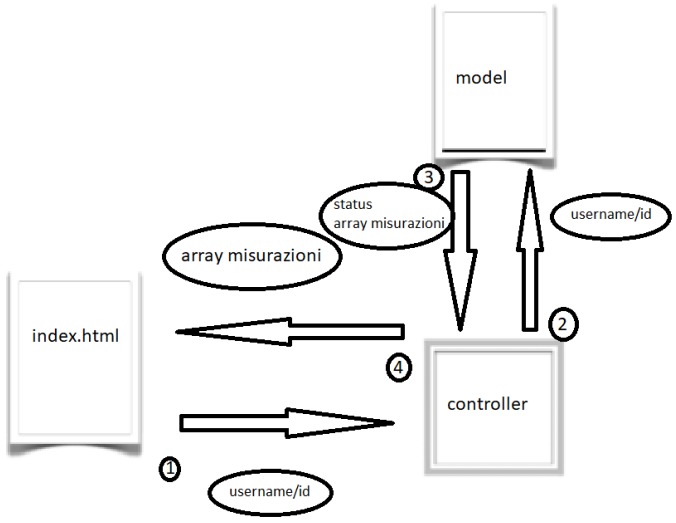


Login / Register

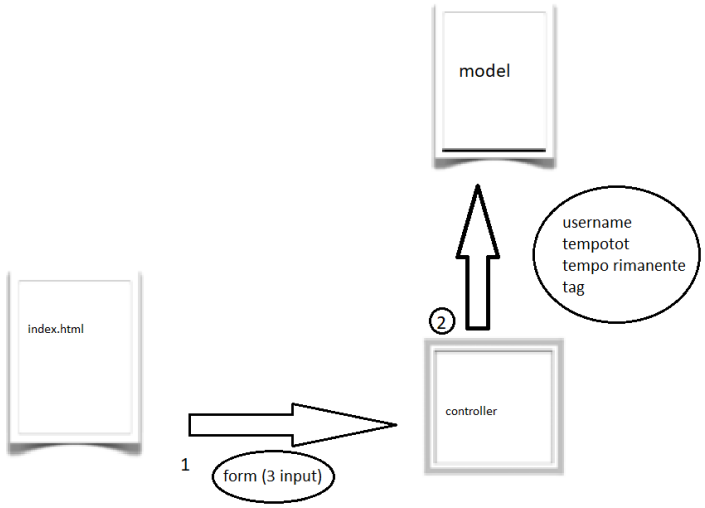
3. Architettura



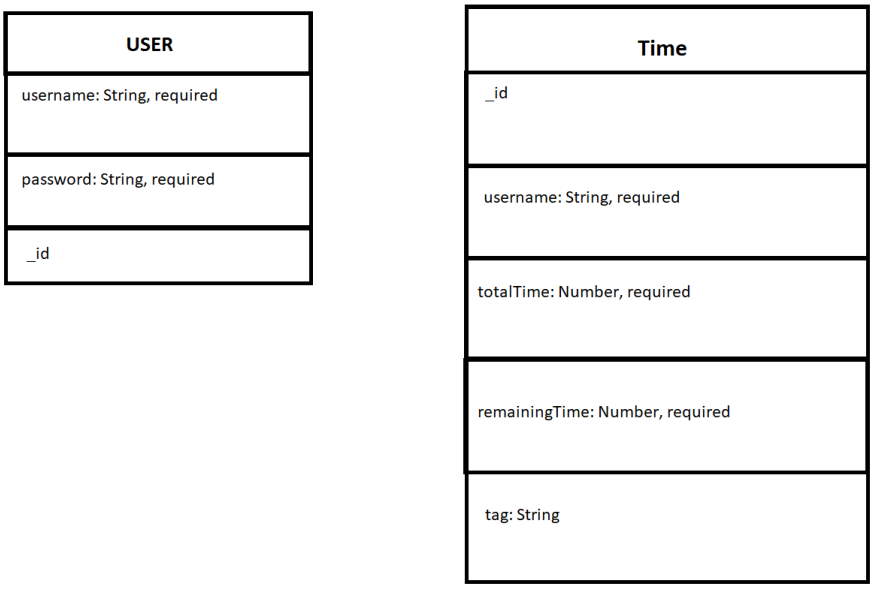
GET


model
3
status
array misurazioni
username/id
array misurazioni
index.html
2
4
controller
1
username/id

POST


model
username
tempotot
tempo rimanente
tag
index.html
2
controller
1
form (3 input)

3.3


USER
username: String, required
password: String, required
_id
Time
_id
username: String, required
totalTime: Number, required
remainingTime: Number, required
tag: String

4. Codice

HTML

```
<div class="lefthalf-screen">
    <div class="timer-container">
        <div class="clock">
            <img src="static/images/ios_clock.svg" alt="Timer" class="clock-image">
            <div class="clock-background">
                <div class="mask-left" id="mask-left"></div>
                <div class="mask-right" id="mask-right"></div>
                <div class="mark-hand" id="mark-hand"></div>
            </div>
            <div class="center-circle"></div>
        </div>
    </div>

    <div class="time-form-container">
        <form id="formTimer" action="/datapost" method="post">
            <div class="inputs-form">
                <div class="input-group input-group-lg">
                    <label for="time-input" class="input-group-text">Tempo</label>
                    <input type="number" id="time-input" class="time-input form-control" value=30 min=1 max=60>
                </div>

                <div class="input-group input-group-lg">
                    <label for="tag-input" class="input-group-text">Tag</label>
                    <input type="text" id="tag-input" class="form-control">
                </div>

                <input type="hidden" id ="remaining-time">
            </div>

            <div class="buttons-form">
                <button id="start-timer" type="button" class="btn btn-primary" onclick="LoadTimer()">Inizia Timer</button>
                <button type="submit" id="submit-time" class="btn btn-danger">Stop</button>
            </div>
        </form>
    </div>
</div>
```

```
<div class="form-container">
    <form id="formLogin" action="/login" method="post">

        <input class="form-control" type="text" id="username" placeholder="Username">
        <input class="form-control" type="password" id="password" placeholder="Password" autocomplete="on">

        <button type="submit" class="form-button-login">Login</button>
    </form>
</div>
```

CSS

```css
.clock-background {
    overflow: hidden;
    position: absolute;
    background-color: white;
    border-radius: 50%;
    height: 75%;
    width: 75%;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
}

.clock-background > div[class^='mask'] {
    background-color: red;
    overflow: hidden;
    position: absolute;
    top: 0;
    bottom: 0;
}
.mask-left {
    height: 100%;
    left: 0;
    right: 50%;
    transform-origin: 100% 50%;
}
.mask-right {
    height: 100%;
    left: 50%;
    right: 0;
    transform-origin: 0% 50%;
}

.mark-hand {
    background-color: black;
    width: 2%;
    height: 100%;
    overflow: hidden;
    position: absolute;
    top: 49%;
    left: 49%;
    transform-origin: top center;
    transform: rotate(180deg);

}

.center-circle {
    background-color: black;
    position: absolute;
    border-radius: 50%;
    width: 5%;
    height: 5%;
    left: 50%;
    top: 50%;
    transform: translate(-50%, -50%);
}
```

Node.js

```
app.use(express.static(application_root));
app.use(express.json());


mongoose.connect('mongodb://127.0.0.1:27017/MongoDB', function(err) {
    if(err) {
        console.log('connection error', err);
    } else {
        console.log('connection successful');
    }
});

app.get('/', function (req, res) {
    console.log("Device connecting");
    res.sendFile(application_root + 'index.html', {});
});


app.get('/register', function (req, res) {
    res.sendFile(application_root + 'register.html', {});
});
app.get('/login', function (req, res) {
    res.sendFile(application_root + 'login.html', {});
});

app.post('/register', fixturesUser.register);

app.post('/login', fixturesUser.login);


app.get('/dataget/:username', fixturesTime.GetTimesUser);

app.get('/dataget/:username/:tag', fixturesTime.GetTimesUserTag);

app.get('/deleteTime/:uid', fixturesTime.DeleteOneTime);
```