IoT Fall Detection System

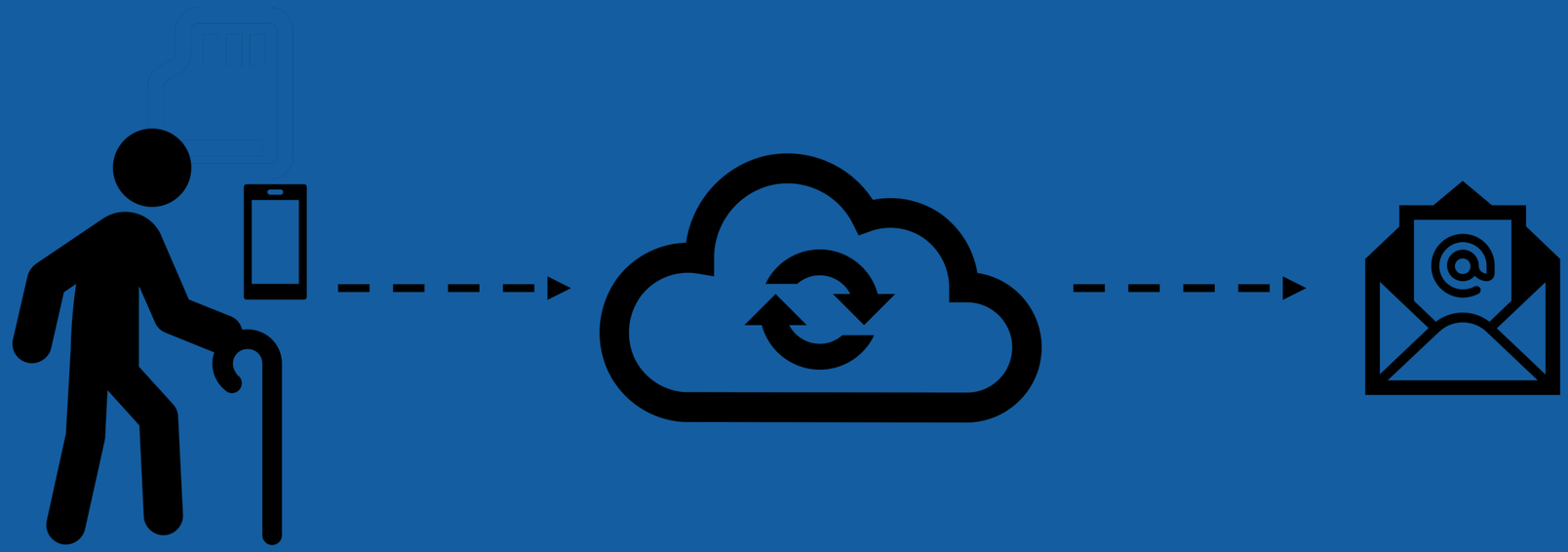
Pervasive Computing e Servizi Cloud

23/01/2024 - Anna Solera

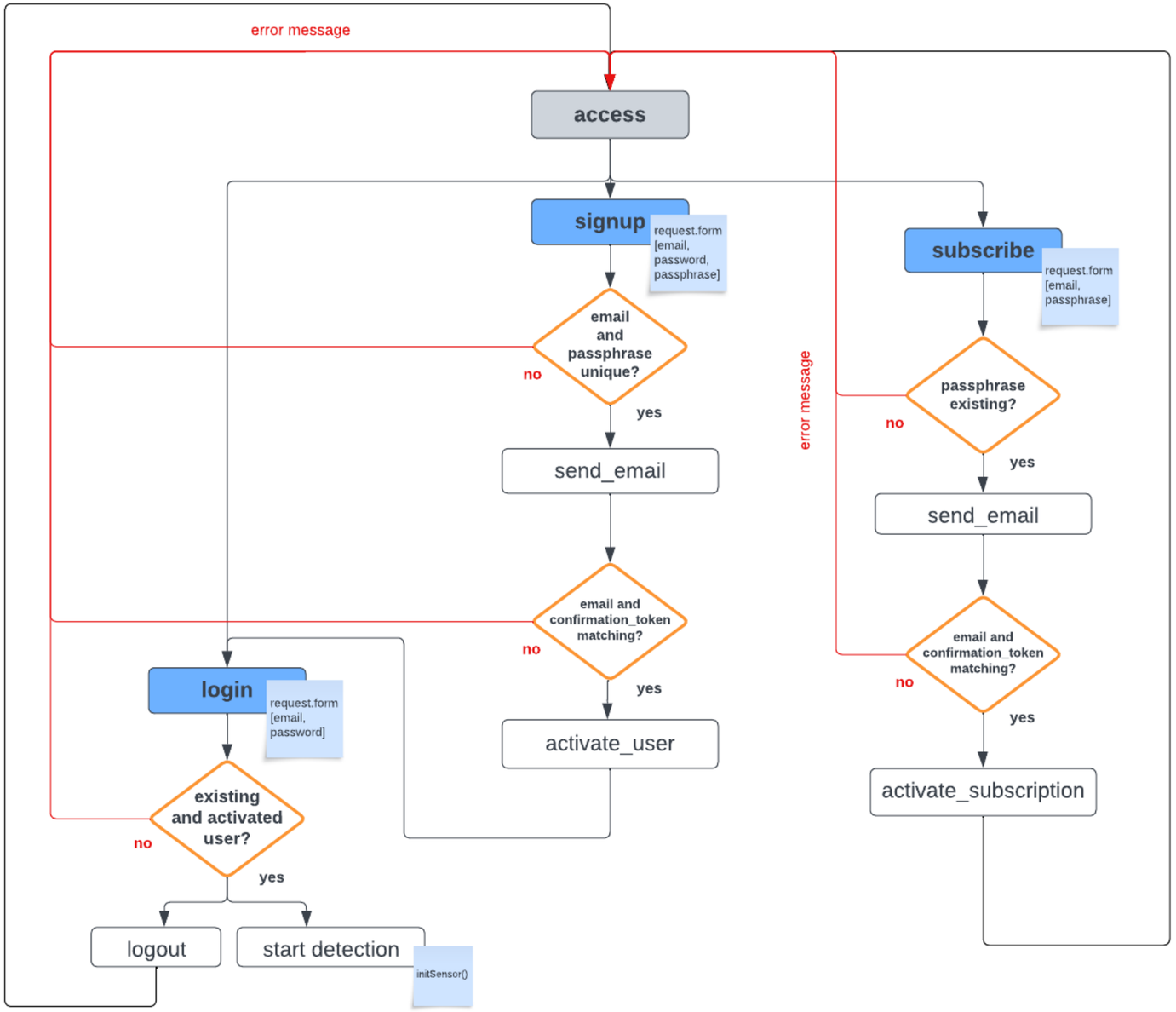
Indice

Architettura del Sistema IoT

Il Sistema IoT di Fall Detection monitora il movimento degli user (in termini di accelerazione) e, quando rileva la caduta di un user, sulla base di una determinata logica, grazie al server cloud, notifica gli utenti sottoscritti a tale user tramite mail.



Scenari di Accesso al servizio



USERS

- id
- email
- password
- passphrase
- confirmation_token
- activated

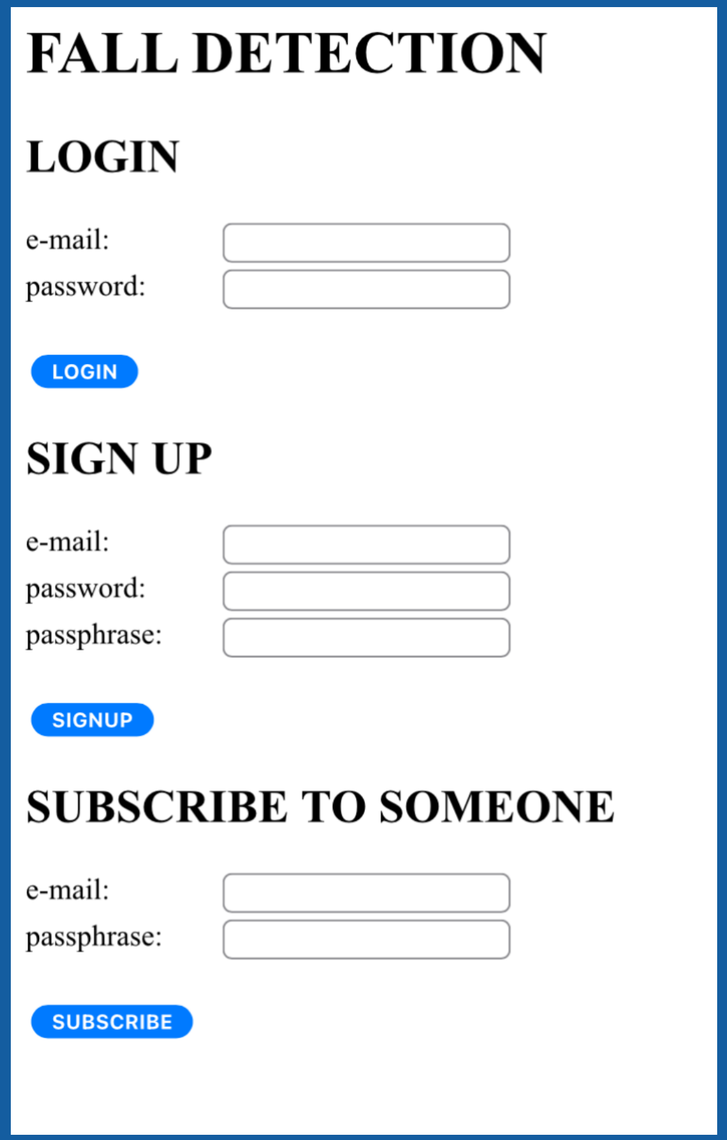
PRIMARY KEY id
UNIQUE email, passphrase

SUBSCRIPTIONS

- id_users
- email
- confirmation_token
- activated

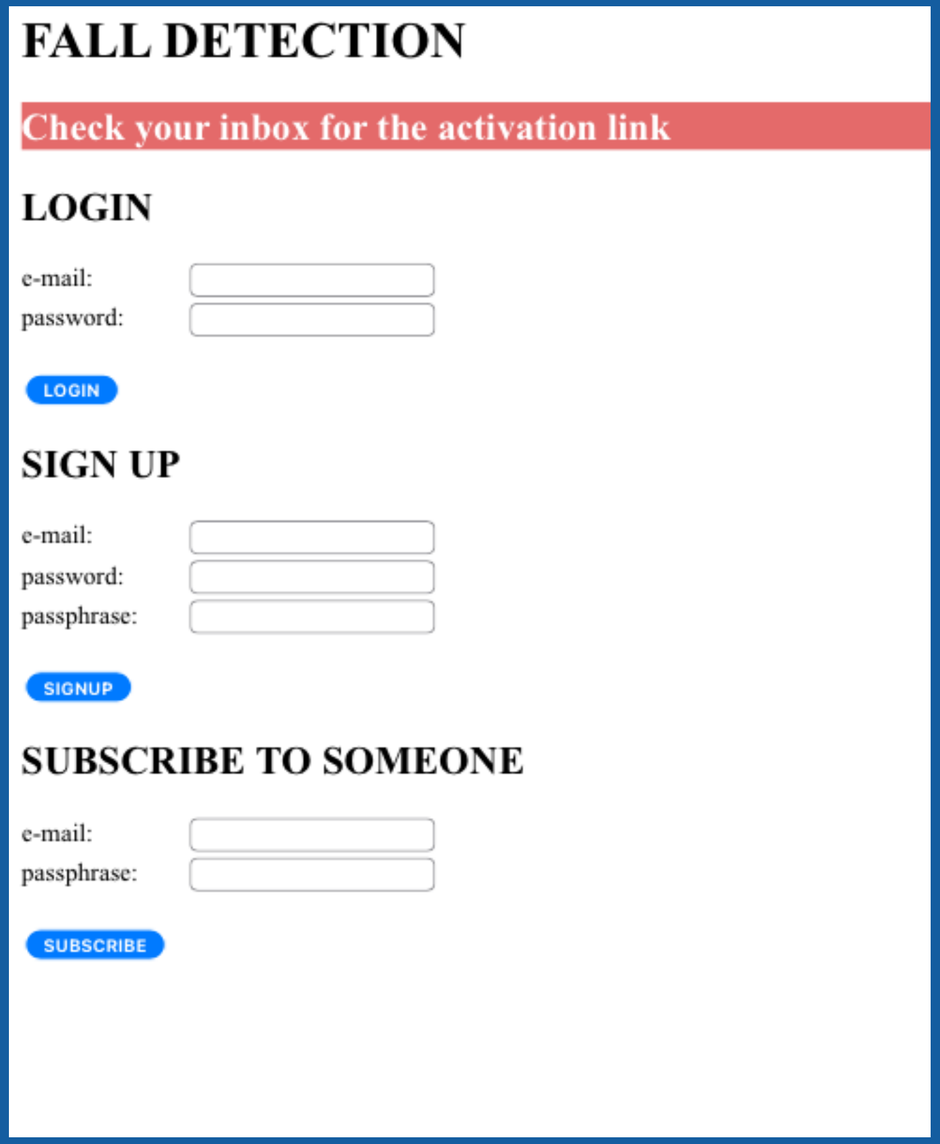
PRIMARY KEY (id_users, email)
FOREIGN KEY id_users = USERS.id

Scenari di Accesso al servizio



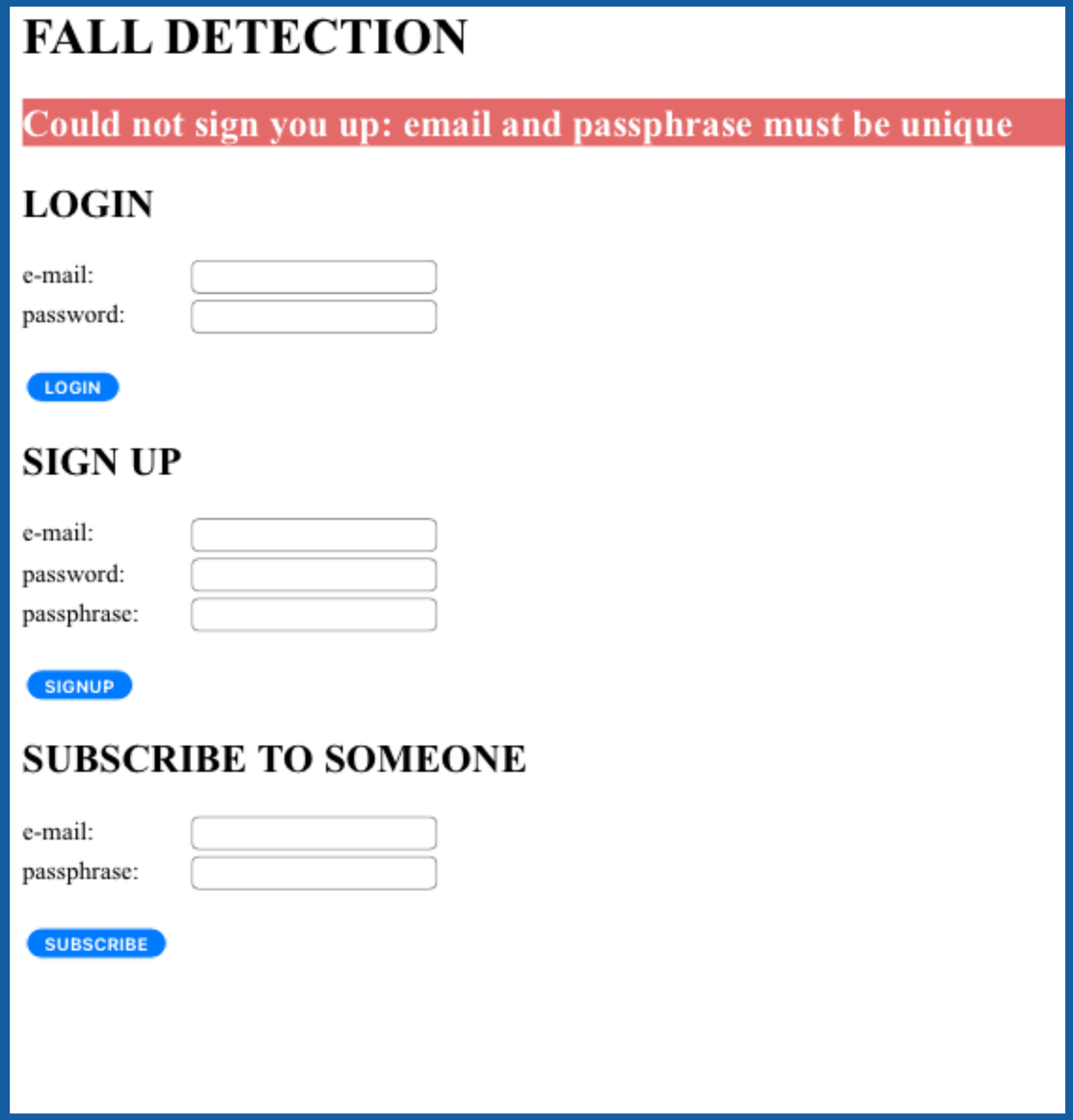

```python
@app.route("/")
def entry_point():
    return redirect(url_for("access"))

@app.route("/access")
def access():

    if "email" in session:
        return redirect(url_for("main", email=session["email"]))

    return render_template("access.html")

@app.route("/main/<email>")
def main(email):
    if "email" not in session:
        return redirect(url_for("access"))

    if email != session["email"]:
        return redirect(url_for("main", email=session["email"]))

    return render_template("index.html", passphrase=session["passphrase"])
```

Scenari di Accesso al servizio




```python
@app.route("/")
def entry_point():
    return redirect(url_for("access"))

@app.route("/access")
def access():

    if "email" in session:
        return redirect(url_for("main", email=session["email"]))

    return render_template("access.html")

@app.route("/main/<email>")
def main(email):
    if "email" not in session:
        return redirect(url_for("access"))

    if email != session["email"]:
        return redirect(url_for("main", email=session["email"]))

    return render_template("index.html", passphrase=session["passphrase"])
```

Scenari di Accesso al servizio




```python
@app.route("/")
def entry_point():
    return redirect(url_for("access"))

@app.route("/access")
def access():

    if "email" in session:
        return redirect(url_for("main", email=session["email"]))

    return render_template("access.html")

@app.route("/main/<email>")
def main(email):
    if "email" not in session:
        return redirect(url_for("access"))

    if email != session["email"]:
        return redirect(url_for("main", email=session["email"]))

    return render_template("index.html", passphrase=session["passphrase"])
```

Scenari di Accesso al servizio – Sign up

```python
@app.route("/signup", methods=["POST"])
def signup():

    cnxn = init_db_connection()
    cursor = cnxn.cursor()

    email = request.form['email'].lower()
    password = request.form['password']
    passphrase = request.form['passphrase']

    # genera stringa random per ogni riga della tabella users nel database
    confirmation_token = "".join(random.choices(string.ascii_lowercase, k=20))

    try:
        query = f"INSERT INTO users (email, password, passphrase, confirmation_token) VALUES ('{email}', '{password}', '{passphrase}', '{confirmation_token}')"
        cursor.execute(query)
        cnxn.commit()
    except:
        return render_template("access.html", message="Could not sign you up: email and passphrase must be unique")

    # mando email con link che contiene email e confirmation token
    confirmation_link = f"{request.base_url}/activate_user/{email}/{confirmation_token}"
    message = f"Please click on the following link to activate your account: {confirmation_link}"
    send_email(email=email, subject="FALL DETECTION: confirm your account", message=message)

    return render_template("access.html", message="Check your inbox for the activation link")
```

```python
def send_email(email, subject, message):

    msg = EmailMessage()
    msg.set_content(message)
    msg['Subject'] = subject
    msg['From'] = "Fall detection service"
    msg['To'] = email

    with smtplib.SMTP_SSL('smtp.gmail.com', 465) as s:
        pwd = open("email.txt").read().strip()
        s.login("annasolera98@gmail.com", pwd)
        s.send_message(msg)
```

Scenari di Accesso al servizio – Sign up

```python
@app.route("/activate_user/<email>/<confirmation_token>")
def activate_user(email, confirmation_token):
    # controlla nel db per email e random_number
    cnxn = init_db_connection()
    cursor = cnxn.cursor()
    query = f"SELECT id, passphrase FROM users WHERE email = '{email}' AND confirmation_token = '{confirmation_token}'"
    cursor.execute(query)
    query_result = cursor.fetchall()

    if len(query_result) == 0:
        return render_template("access.html", message="Could not activate the user")

    # se compatibili, allora cambia il db mettendo il campo valid a uno
    id, passphrase = query_result[0]
    query = f"UPDATE users SET activated = 1 WHERE id = {id}"
    print(query)
    cursor.execute(query)
    cnxn.commit()

    # renderizza login con messaggio utente attivo, puoi fare il login
    session["id"] = id
    session["email"] = email
    session["passphrase"] = passphrase
    return redirect(url_for('main', email=email))
```

Scenari di Accesso al servizio – Login

```python
@app.route("/login", methods=["POST"])
def login():

    email = request.form["email"].lower()
    password = request.form["password"]

    cnxn = init_db_connection()
    cursor = cnxn.cursor()
    query = f"SELECT id, passphrase, activated FROM users WHERE email = '{email}' AND password = '{password}'"
    cursor.execute(query)
    query_result = cursor.fetchall()

    # account non trovato
    if len(query_result) == 0:
        return render_template("access.html", message="Email / password not matched")

    # account trovato ma non attivato
    id, passphrase, activated = query_result[0]
    if activated == 0:
        return render_template("access.html", message="Check your inbox for the activation link")

    # OK, ci possiamo loggare
    session["id"] = id
    session["email"] = email
    session["passphrase"] = passphrase
    return redirect(url_for('main', email=email))
```

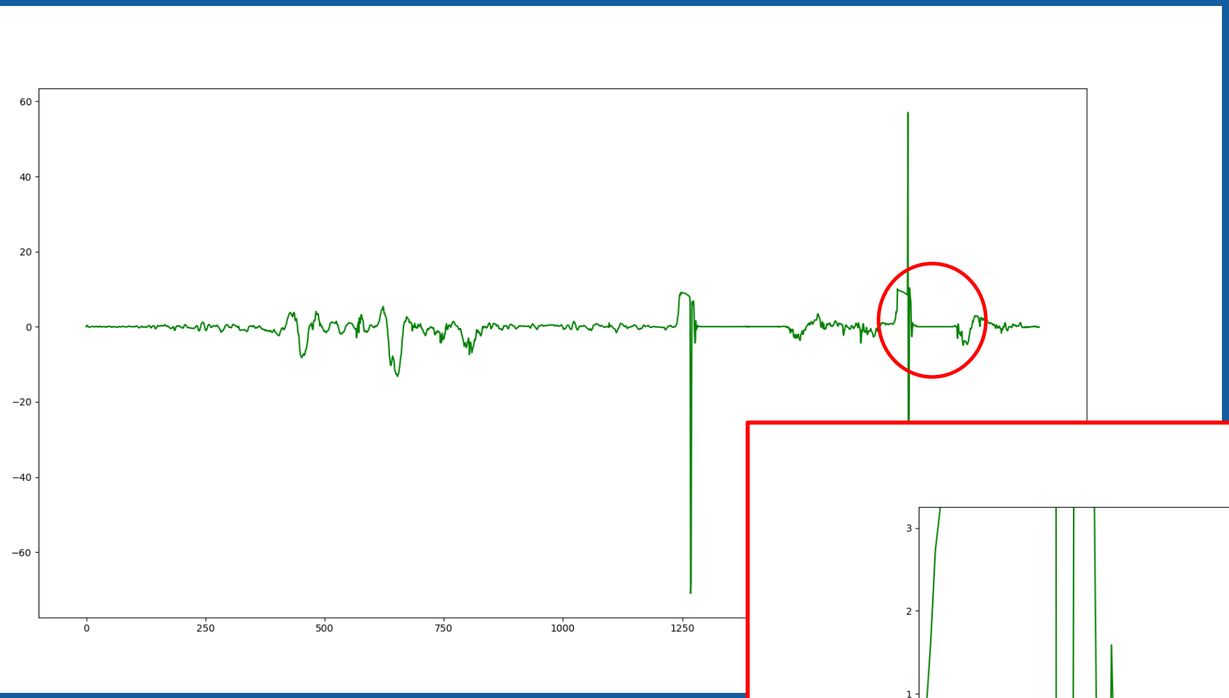
Scenari di Accesso al servizio – Subscribe

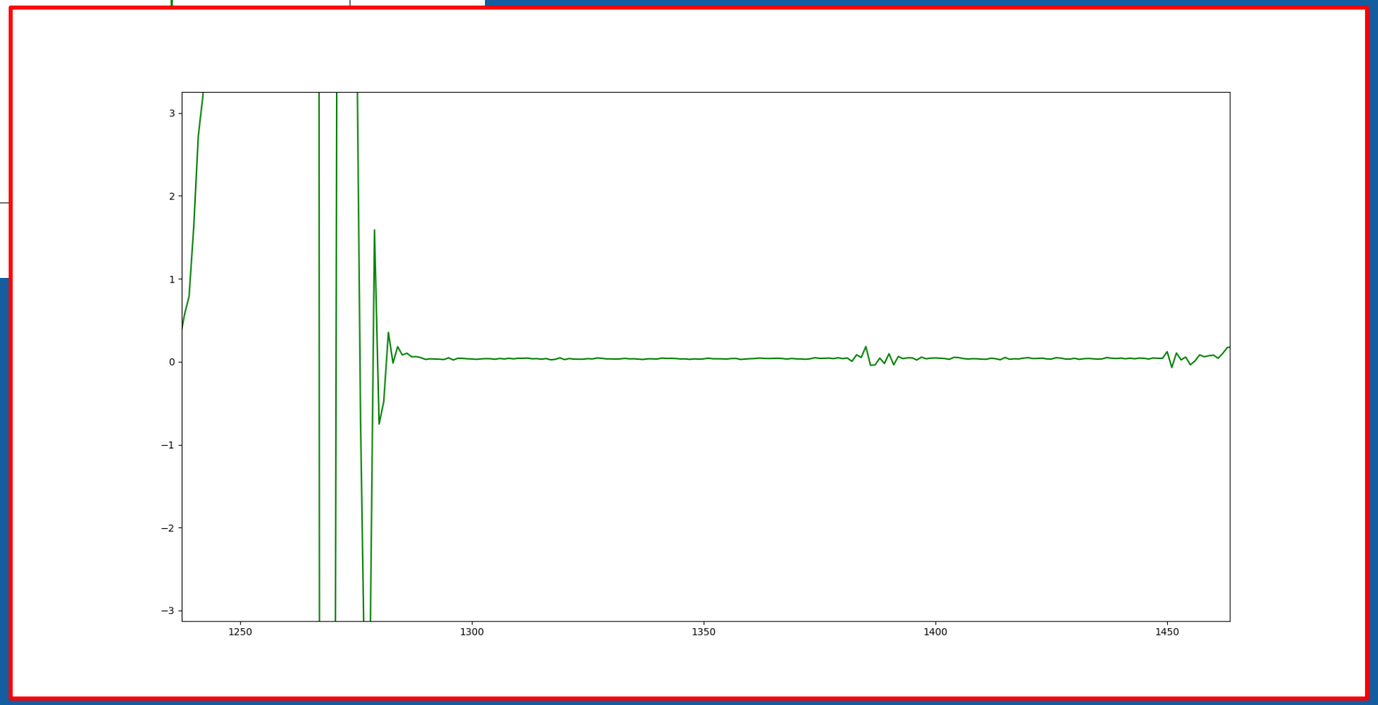
```python
@app.route("/subscribe", methods=["POST"])
def subscribe():

    email = request.form["email"].lower()
    passphrase = request.form["passphrase"]

    # check if passphrase exists
    cnxn = init_db_connection()
    cursor = cnxn.cursor()
    query = f"SELECT id, email FROM users WHERE passphrase = '{passphrase}' AND activated = 1"
    cursor.execute(query)
    query_result = cursor.fetchall()

    if len(query_result) == 0:
        return render_template("access.html", message="The passphrase does not point to any active user")

    id_users, email_users = query_result[0]

    # genera stringa random per ogni riga della tabella subscription nel database
    confirmation_token = "".join(random.choices(string.ascii_lowercase, k=20))

    try:
        query = f"INSERT INTO subscriptions (id_users, email, confirmation_token) VALUES ({id_users}, '{email}', '{confirmation_token}')"
        cursor.execute(query)
        cnxn.commit()
    except:
        return render_template("access.html", message="You are already subscribed to the user.")

    # mando email con link che contiene email e confirmation token
    confirmation_link = f"{request.url_root}/activate_subscription/{email}/{confirmation_token}"
    message = f"Please click on the following link to activate your subscription to {email_users}: {confirmation_link}"
    send_email(email=email, subject="FALL DETECTION: confirm your subscription", message=message)

    return render_template("access.html", message="Check your inbox for the link to activate your subscription")
```

Scenari di Accesso al servizio – Subscribe

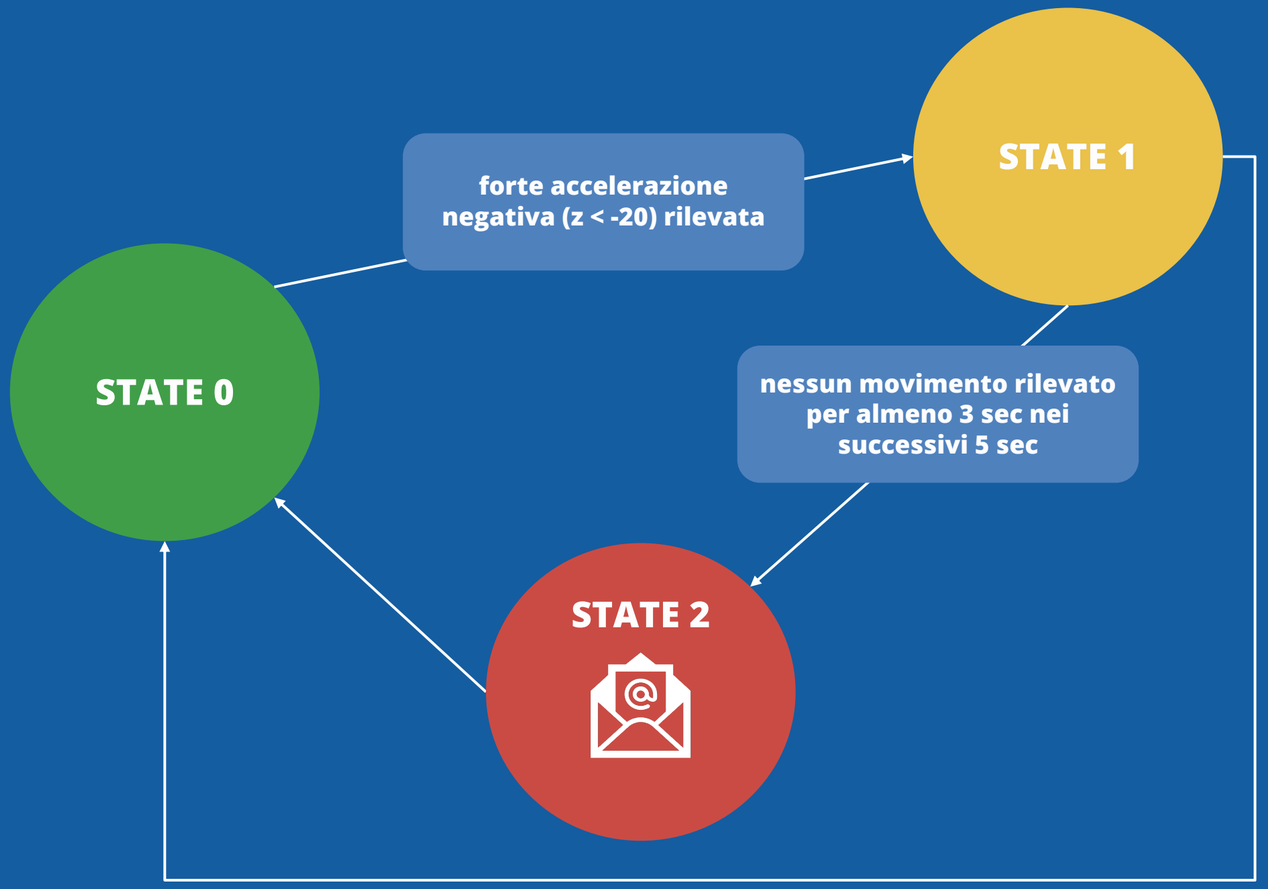
```python
@app.route("/activate_subscription/<email>/<confirmation_token>")
def activate_subscription(email, confirmation_token):
    # controlla nel db per email e confirmation_token
    cnxn = init_db_connection()
    cursor = cnxn.cursor()
    query = f"SELECT id_users FROM subscriptions WHERE email = '{email}' AND confirmation_token = '{confirmation_token}'"
    cursor.execute(query)
    query_result = cursor.fetchall()

    if len(query_result) == 0:
        return render_template("access.html", message="Could not activate the subscription")

    # se compatibili, allora cambia il db mettendo il campo valid a uno
    id_users = query_result[0][0]
    query = f"UPDATE subscriptions SET activated = 1 WHERE id_users = {id_users} AND email = '{email}'"
    cursor.execute(query)
    cnxn.commit()

    # renderizza login con messaggio utente appropriato
    return render_template("access.html", message="You are subscribed!")
```

Logica di rilevazione cadute





Logica di rilevazione cadute




```
// STATE 0: detect strong negative acceleration for even just one frame
if (current_state == 0 & z < -20) {
    state_1_for_ms = 0;
    state_1_no_motion_ms = 0;
    current_state = 1;
    state_1_last_timestamp = Date.now();

// STATE 1: detect no motion for more than 2 sec in the following 3 sec
} else if (current_state == 1) {
    delta_ms = Date.now() - state_1_last_timestamp;
    state_1_last_timestamp = Date.now();

    state_1_for_ms += delta_ms;

    if (abs(z) < 2) {
        state_1_no_motion_ms += delta_ms;
    }

    if (state_1_for_ms > 5000) {
        if (state_1_no_motion_ms > 3000) {
            current_state = 2;
        } else {
            current_state = 0;
        }
    }

// STATE 2: send email if not already sent in the last 10 seconds
} else if (current_state == 2) {
    current_state = 0;
    if (Date.now() - last_event_recorded > 10000) {
        last_event_recorded = Date.now();

        $.post("/send_event", {}, function(data, status) {
            $("#output").html("FALL DETECTED, email sent to: " + data)
        })
    }
}
```

Logica di rilevazione cadute


```python
@app.route("/send_event", methods=["POST"])
def upload_data():

    cnxn = init_db_connection()
    cursor = cnxn.cursor()
    query = f"SELECT email FROM subscriptions WHERE id_users = '{session['id']}' AND activated = 1"
    cursor.execute(query)
    query_result = cursor.fetchall()

    email_sent_to = []
    for row in query_result:
        message = f"FALL DETECTED FOR USER {session['email']}"
        send_email(row[0], subject=message, message=message)
        email_sent_to.append(row[0])

    return ", ".join(email_sent_to)
```

```javascript
// STATE 0: detect strong negative acceleration for even just one frame
if (current_state == 0 & z < -20) {
    state_1_for_ms = 0;
    state_1_no_motion_ms = 0;
    current_state = 1;
    state_1_last_timestamp = Date.now();

// STATE 1: detect no motion for more than 2 sec in the following 3 sec
} else if (current_state == 1) {
    delta_ms = Date.now() - state_1_last_timestamp;
    state_1_last_timestamp = Date.now();

    state_1_for_ms += delta_ms;

    if (abs(z) < 2) {
        state_1_no_motion_ms += delta_ms;
    }

    if (state_1_for_ms > 5000) {
        if (state_1_no_motion_ms > 3000) {
            current_state = 2;
        } else {
            current_state = 0;
        }
    }

// STATE 2: send email if not already sent in the last 10 seconds
} else if (current_state == 2) {
    current_state = 0;
    if (Date.now() - last_event_recorded > 10000) {
        last_event_recorded = Date.now();

        $.post("/send_event", {}, function(data, status) {
            $("#output").html("FALL DETECTED, email sent to: " + data)
        })
    }
}
```

Grazie per l’attenzione!