Coradin Emanuele
IIS Euganeo
classe : 5BI

SISTEMA DI MONITORAGGIO

13 giugno 2020

Premessa medica

L'infezione covid 19 presenta una sintomatologia simile a quella dell'influenza andando a colpire principalmente il sistema polmonare. Quindi, secondo la comunità scientifica, uno dei primi segnali che possono indicare una prossima manifestazione della malattia è la discesa dell'indice SP02 sotto il 92%. SP02 indica la percentuale di emoglobina satura di ossigeno rispetto a quella presente nel sangue e normalmente il suo valore si aggira intorno al 98-99%. Grazie ad un saturimetro digitale è possibile rilevarlo in modo rapido e comodo (il tempo richiesto è inferiore ai 30 secondi), insieme ad una serie di altre informazioni utili per il medico (frequenza cardiaca, frequenza respiratoria etc.).

Proposta

Vista la premessa medica, si propone di realizzare un sistema informatico nazionale, in modo da monitorare i pazienti in isolamento domiciliare e le persone ad alto rischio e garantire una diagnosi tempestiva.

Obiettivi

1. **Monitorare i pazienti in isolamento domiciliare.**
2. **Favorire una diagnosi precoce nelle persone più a rischio.**
3. **Raccogliere dati statistici utili per migliorare le tecniche di diagnosi (Machine Learning), analizzare e mappare le aree di contagio.**

Descrizione generale

Figura 1: Saturimentro digitale

Per la realizzazione di questo progetto si propone di fornire un saturimetro digitale ad ogni paziente. I dispositivi saranno in grado di inviare i dati raccolti allo smartphone o al tablet del paziente tramite Bluetooth. verranno raccolti da un'apposita applicazione precedentemente scaricata presso qualsiasi store digitale (Google playstore, APP store…), la quale provvederà anche a inviarli al server centrale tramite Wi-Fi o rete mobile. Nel caso in cui un paziente non dovesse disporre di un dispositivo mobile, esso verrà fornito in usufrutto dallo Stato. Se non dovesse disporre nemmeno di una rete Wi-Fi o di una connessione alla rete mobile, gli sarà fornita un'apposita schedina SIM.

Per partecipare al programma i medici di base dovranno compilare un form di iscrizione specificando le proprie generalità (vedi paragrafo **database**) e allegando un documento di riconoscimento. I dati verranno quindi confrontati con i database nazionali degli ordini professionali e, in caso di corrispondenza, verranno spedite le credenziali di accesso al sistema tramite posta elettronica certificata.

A questo punto, (previa autenticazione alla pagina di **login**) il medico potrà accedere ad una dashboard online, grazie alla quale sarà in grado di controllare i parametri dei propri pazienti iscritti al progetto. Inoltre potrà accedere al bot Telegram, disponibile all'indirizzo **t.me/CovidMbBot**, avviandolo col comando "/start codiceFiscale password", rispettando ordine e spazi.

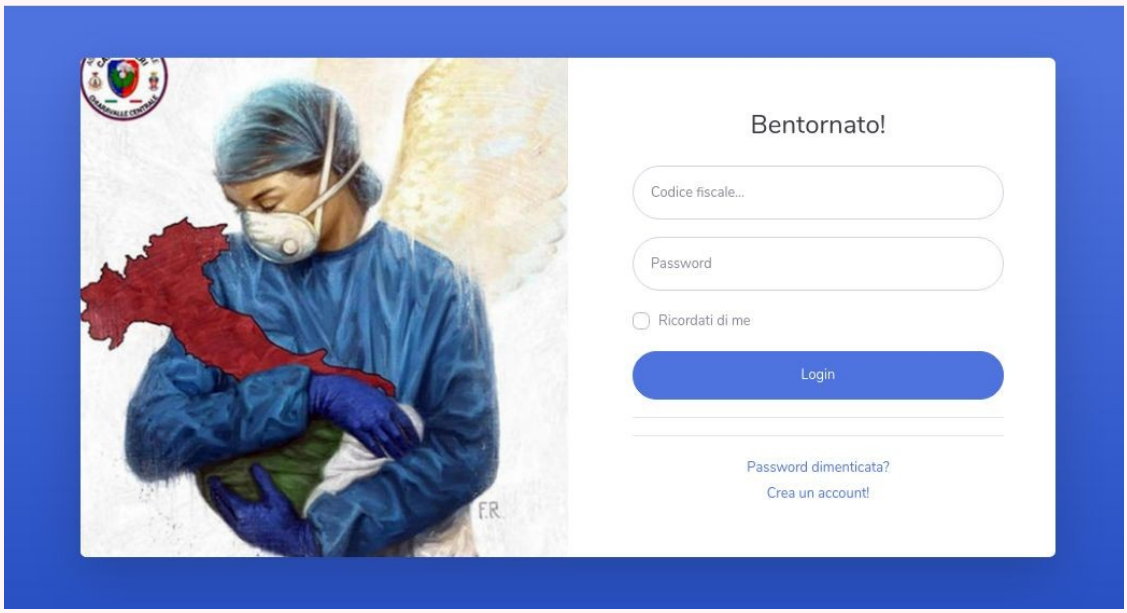


Figura 2: Pagina di login

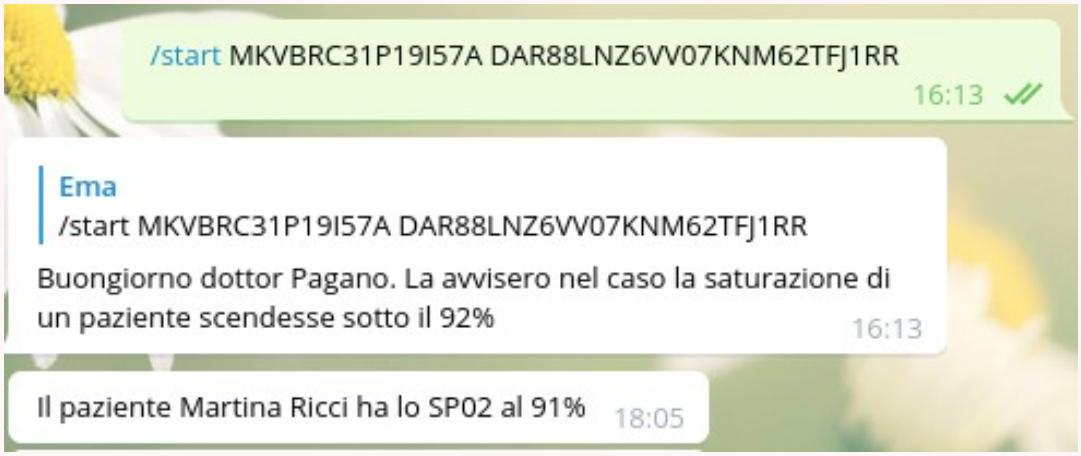


Figura 3: Esempio utilizzo bot Telegram

Per iscrivere un nuovo paziente al progetto, occorre compilare un apposito modulo di richiesta, specificando le sue generalità, allegando un documento di identità valido e inserendo un elenco delle patologie pregresse e in corso. A seguito della verifica dell'identità, infatti, quei dati verranno elaborati da un programma di Machine Learning (**CV-19 index**) che restituirà un indice numerico del rischio di andare in contro ad uno sviluppo grave dell'infezione. Questa informazione servirà come criterio oggettivo per stilare una lista di priorità: in caso non si riescano a soddisfare nel breve termine tutte le richieste, verranno privilegiati i soggetti con indice di vulnerabilità maggiore. Inoltre, questo dato può essere utile anche agli stessi medici di base per le loro valutazioni.

Una volta calcolato l'indice di vulnerabilità verrà generata automaticamente una password robusta di 16 caratteri e i dati verranno inseriti nel database.

Le credenziali di accesso verranno quindi inviate via PEC al medico di base, che provvederà a impostare l'app insieme al paziente. Le operazioni da svolgere sono semplici: avviarla, fornirle i permessi necessari ed inserire username e password. A questo punto il paziente non dovrà far altro che avviarla, attivare il bluetooth, il wi-fi/rete mobile sul dispositivo, accendere il saturimetro ed effettuare le misurazioni secondo la frequenza indicato dal medico.

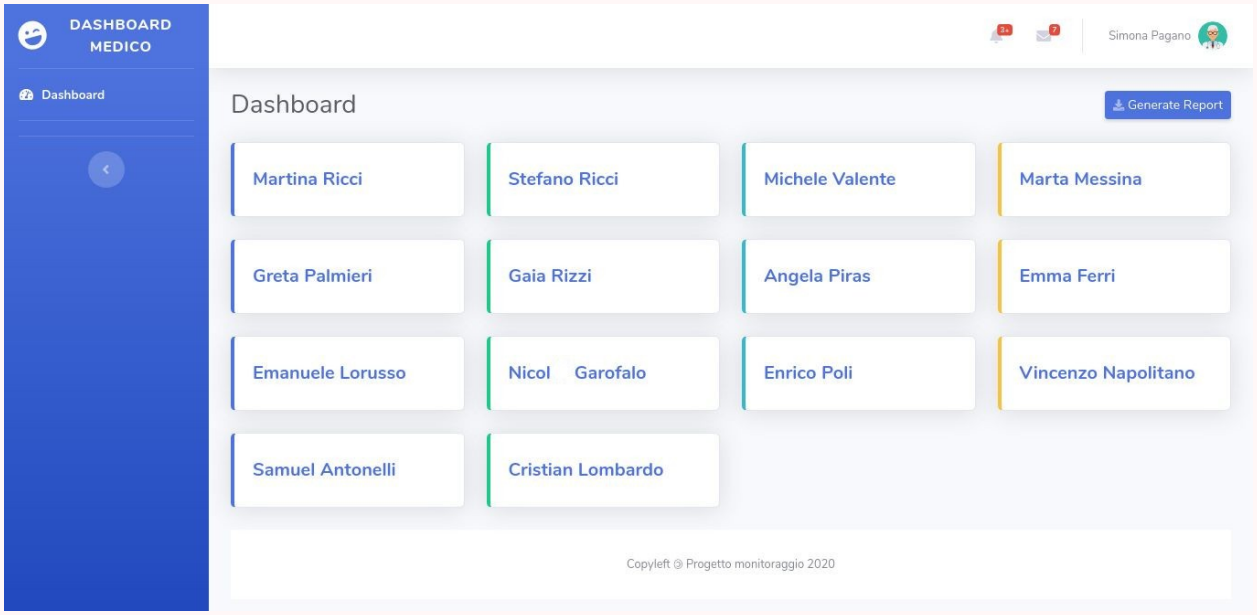


Figura 4: Dashboard medico, schermata principale

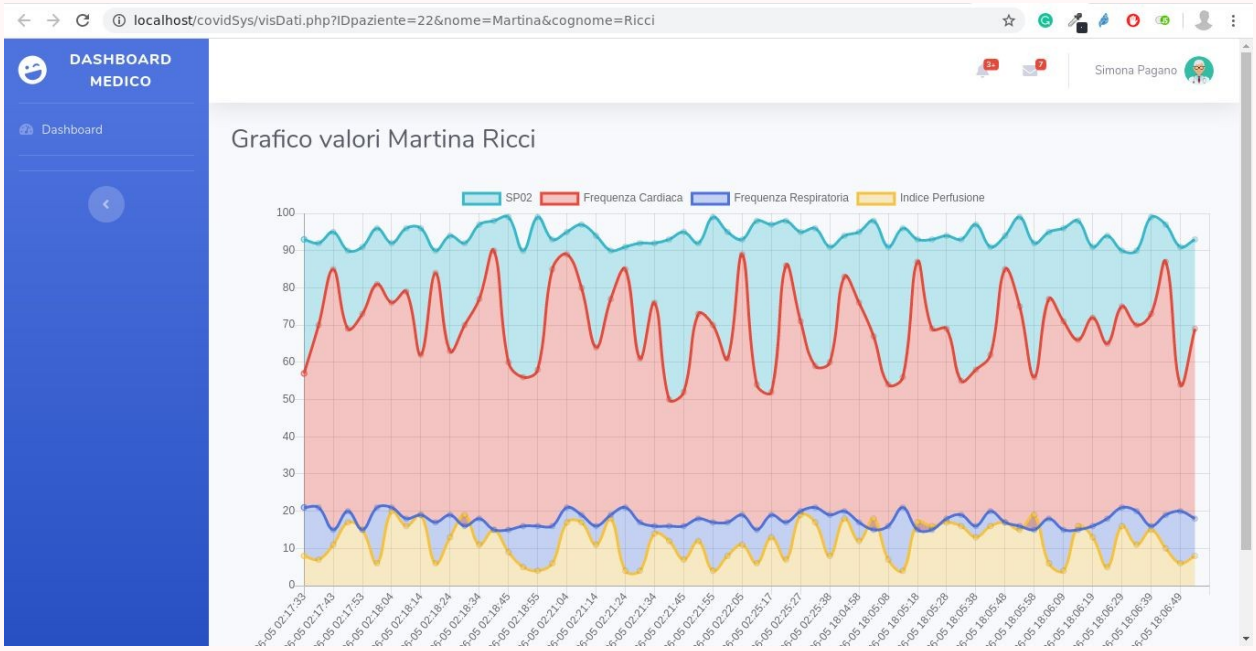


Figura 5: Esempio schermata di visualizzazione dei valori

I dati verranno infine inviati al centro di controllo, registrati nel database, e resi accessibili al dottore tramite dashboard. Inoltre, nel caso in cui l'indice SP02 di un paziente dovesse risultare minore o uguale a 92 verrà inviato un messaggio di allarme al suo medico di base tramite bot telegram.

Nel caso in cui un paziente dovesse essere ricoverato in terapia intensiva o morire, il medico provvederà a segnalarlo tramite il portale e i dati confluiranno nel database.

Essi potranno poi essere impiegati per allenare la rete neurale che calcola l'indice di vulnerabilità e sfruttati per altre indagini mediche e statistiche.

Trattazione tecnica

Protocolli e linguaggi

- CRITTOGRAFIA:
  - **RSA,**
  - **AES,**
  - **HASH.**

- LINGUAGGI:
  - **PYTHON,**
  - **PHP (e SQL).**

- COMUNICAZIONE:
  - **HTTPS,**
  - **MQTT,**
  - **BLUETOOTH,**
  - **PEC.**

INFRASTRUTTURA DI RETE

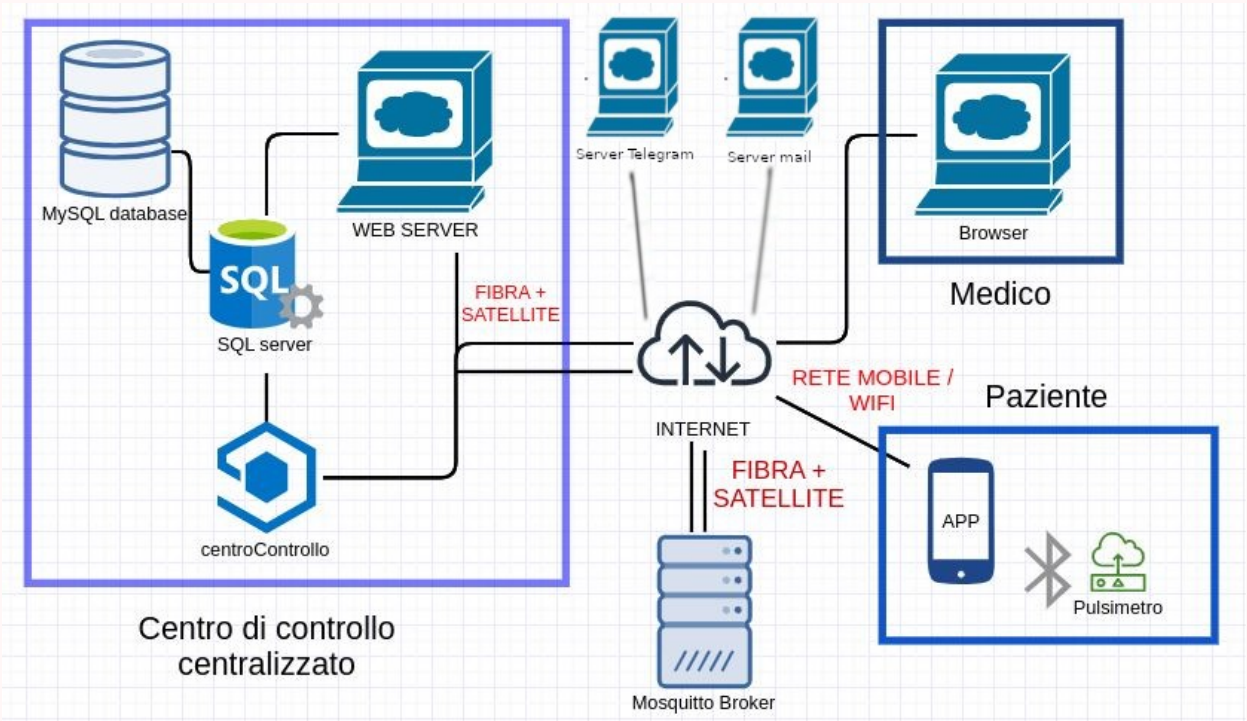


Figura 6: Schema infrastruttura di rete

Come accennato in precedenza, la comunicazione tra le app installate nei dispositivi dei pazienti e il centro di controllo avviene grazie al protocollo MQTT.

ALLOCAZIONE SERVER E SCRIPT

Per la PEC ci si potrebbe appoggiare al sistema di Poste Italiane, quindi l'alloggiamento del server mail è affidato ad un ente esterno, proprio come il server Telegram.

Il server web, Mosquitto, il database MySQL, gli script python del centro di controllo, della generazione delle chiavi asimmetriche e del CV-19 index potrebbero essere ospitati all'interno di un data center statale (come quelli che fanno capo al consorzio interuniversitario **Cineca**).

In questo modo si ottengono i seguenti **vantaggi**:

- **Economici**: utilizzando strutture già esistenti e di possesso statale non occorre far fronte ad ulteriori costi per confezionare da zero una soluzione in housing, né pagare un ente esterno per utilizzare le sue infrastrutture;
- **Temporali**: l'implementazione sarebbe rapida quasi come se avessi scelto una soluzione in outsourcing;
- **Di sicurezza**: la sicurezza fisica del sistema e la disponibilità dei servizi è garantita dall'ente stesso, che ha già implementato tutte le misure necessarie (server in DMZ, ridondanza dei canali, controllo degli accessi fisici etc.).

Sicurezza dei server e del database

Per quanto riguarda il server mail e il server Telegram la sicurezza è affidata alle stesse imprese, quindi non occorre preoccuparsi personalmente di implementare alcuna soluzione.

Per garantire la sicurezza del database occorre:

- Effettuare periodici backup;
- Implementare le prepared statement in php per impedire attacchi del tipo SQL injection(**login.php**). Al momento (per motivi di tempo) su **visDati.php** utilizzo la funzione php mysqli -> real_escape_string($var) sui dati passati tramite metodi GET e POST, che è comunque una buona soluzione;
- Non stampare eventuali errori riguardanti le query per non rivelare informazioni sulla struttura del database (al momento gli errori vengono stampati per debug);
- Non salvare direttamente le password ma le loro hash (per il momento non accade solo per comodità nello sviluppo e nella presentazione del sistema);
- Creazione dei seguenti account (oltre a quelo di root che **non** deve essere usato negli script di ordinaria amministrazione del sistema):
  - **Medico**, con privilegi:
    - GRANT SELECT ON PAZIENTE (non sul campo password), MEDICO, MISURAZIONE
    - GRANT UPDATE (IDstato) ON PAZIENTE
    - INSERT ON DIAGNOSI
  - **Paziente**, con privilegi:
    - GRANT SELECT ON PAZIENTE
    - GRANT INSERT ON MISURAZIONE

  Ovviamente tali account devono essere protetti da password e non saranno mai utilizzati direttamente dagli utenti, bensì dai vari script (l'account medico nel file **DBconnection.php** e l'account utente in **centroControllo.py**, per esempio);
- Disattivare tutti i servizi non utilizzati (sia nel web server che nel sql server) per ridurre al minimo necessario il numero di porte aperte;

- Mantenere il sistema costantemente aggiornato;

- Utilizzare dei sistemi di Intrusion Detection Systems (IDS) o di Intrusion Prevention Systems (IPS), che lavorano con diversi sistemi di riconoscimento per individuare tempestivamente (e magari contrastare) gli attacchi sul server;
- Installare lo script del centroControllo, il web server e il server sql in host differenti, per impedire che l'accesso ad uno di essi comprometta tutti gli altri.;
- Posizionare tutti i server in DMZ.
- Infine è necessario rendere il sistema conforme al GDPR e alla norma ISO/IEC 27001.

Sicurezza delle comunicazioni

Per quanto riguarda la comunicazione con il bot Telegram non occorre prendere particolari precauzioni perché è lo stesso servizio dell'applicazione di messaggistica che garantisce la sicurezza del canale, mentre per le pagine web visibili dal medico occorre semplicemente adottorrare il protocollo HTTPS. Per quanto riguarda gli script python comunicanti con protocollo MQTT, invece, è più complesso. Per garantire la sicurezza dei dati in passaggio, ho deciso di combinare insieme due soluzioni: a livello di trasporto si sfrutta il protocollo TLS, mentre a livello application si cripta il payload dei messaggi e si calcola l'hash. In questo modo viene garantito un ottimo grado di sicurezza della comunicazione (riservatezza e integrità), pagando con un aumento della potenza di calcolo richiesta. Tuttavia ciò non è un problema, perché la quantità di byte da inviare è contenuta.

Inoltre, mentre su tutti i messaggi viene calcolata la funzione di Hash per il controllo dell'integrità non è necessario che tutti siano criptati, per esempio quando si invia la chiave pubblica RSA lo si può tranquillamente effettuare in chiaro.

Per la generazione delle chiavi vedere il file **crypto.py**.

Più in dettaglio, la comunicazione avviene nel seguente modo:

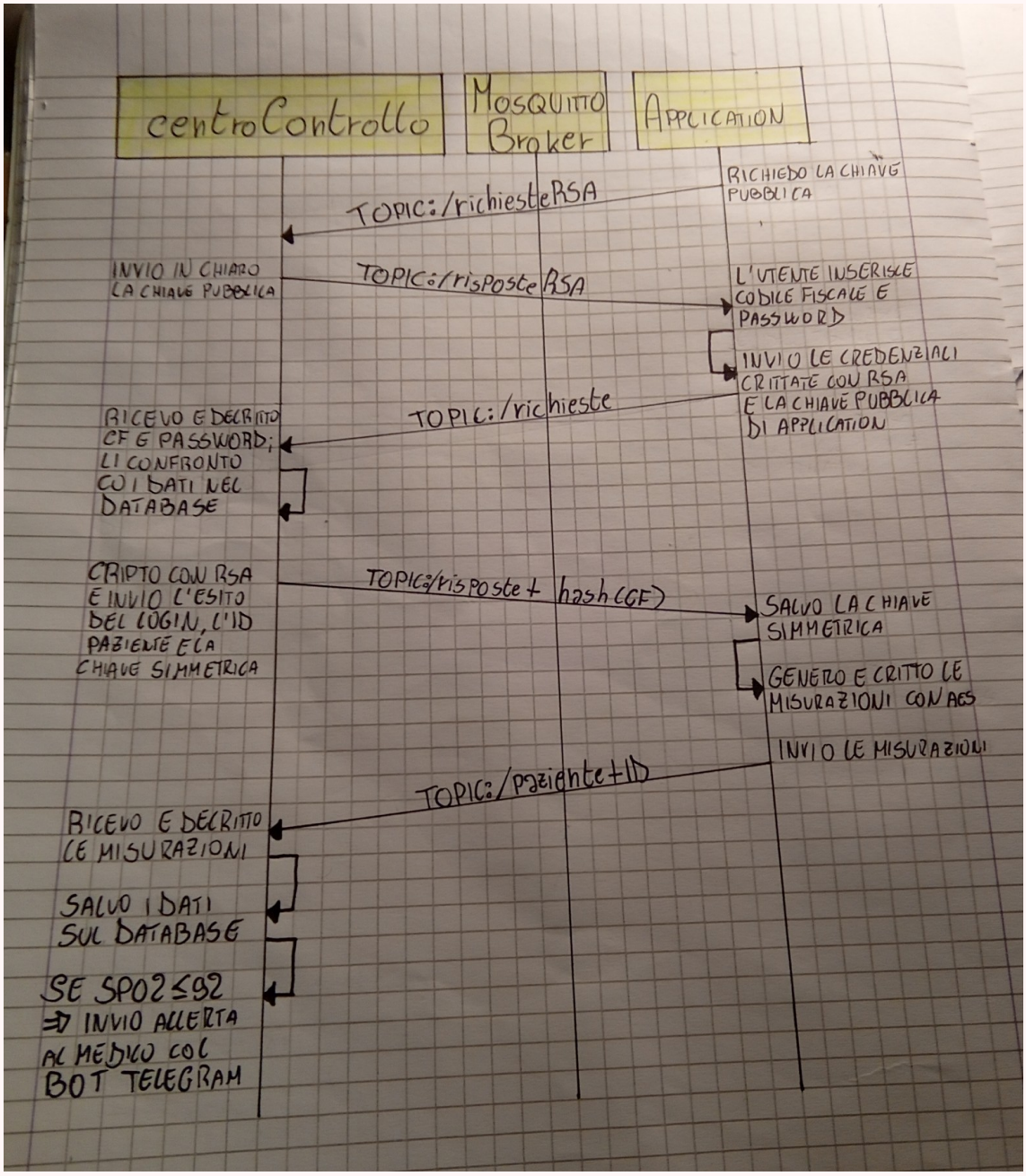


Figura 7: Schema comunicazione MQTT

Estratti di codice più significativi:

```python
#funzione per pubblicare aggiungendo la funzione di hash
def pubblica(msgTopic, msgPayload, broker):
    msgPayload = str(hash(msgPayload)) + " HASH " + msgPayload
    publish.single(msgTopic, msgPayload, hostname=broker)

#funzione per separare la chiave di hash dal payload
def separaMsg(msgPayload):
    coppia =  msgPayload.split(" HASH ")
    if(len(coppia) != 2):
        coppia[0] = -1
        coppia.append("")
    return coppia[0], coppia[1]

#controllo dell'integrita del messaggio
def isUntouched(msgPayload):
    hashSum, carico = separaMsg(msgPayload)
    if(len(carico) != 0 and hash(carico) == int(hashSum)):
        return True
    return False
```

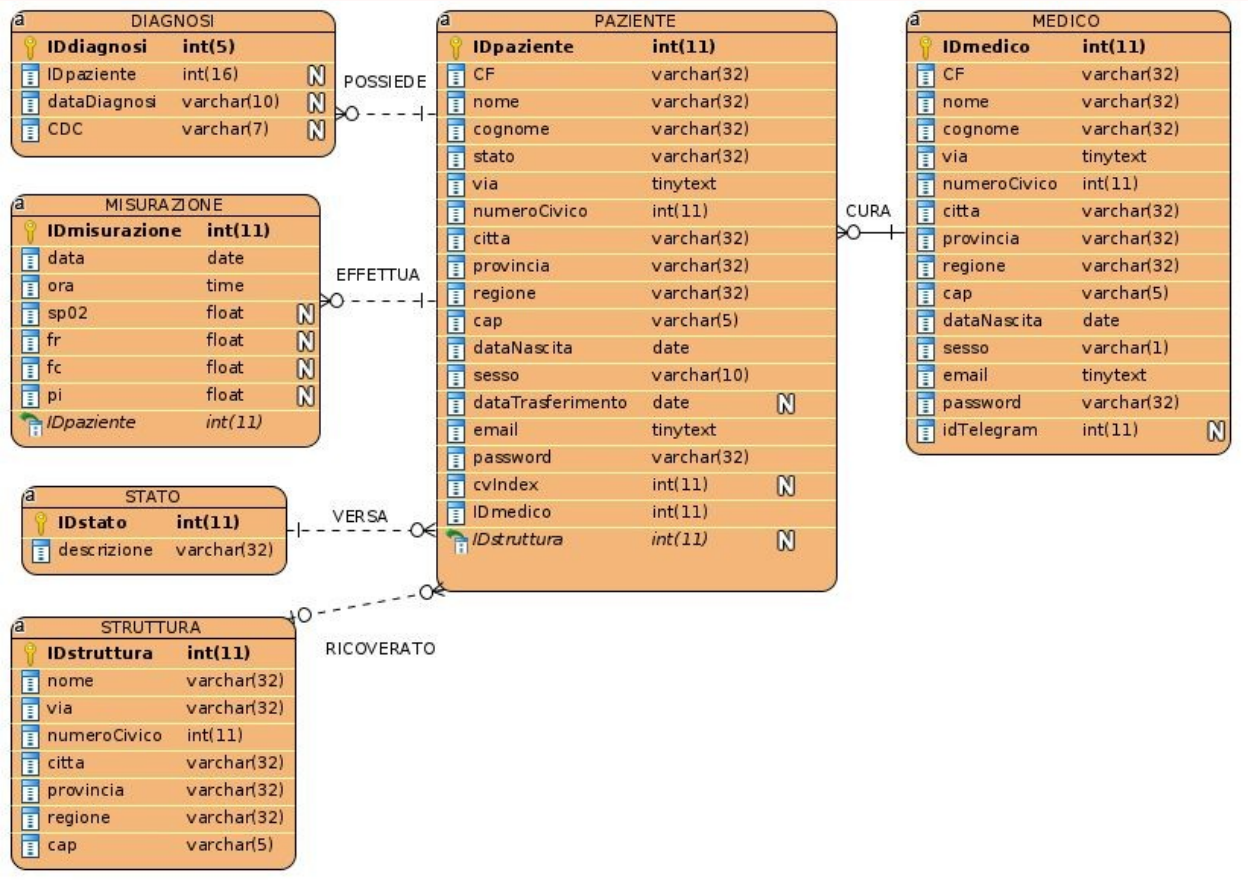
Figura 8: Funzioni per il controllo dell'integrità

```python
#generazione delle chiavi
keyPair = RSA.generate(2048)

#scrivo la chiave privata su file
privf = open('privKeyClient.pem','wb')
privf.write(keyPair.exportKey('PEM'))
privf.close()

#estraggo la chiave pubblica e la scrivo su file
pubKey = keyPair.publickey()
pubf = open('pubKeyClient.pem','wb')
pubf.write(pubKey.exportKey('PEM'))
pubf.close()
```

Figura 9: FIle crypto, generazione chiavi RSA

DATABASE

1. Schema Uml



2. Schema logico

a. STATO(**IDstato**, descrizione);

b. STRUTTURA(**IDstruttura**, nome, via, numeroCivico, citta, provincia, regione, cap);

c. MISURAZIONE(**IDmisurazione**, data, ora, sp02, fr, fc, pi, *Idpaziente*);

d. DIAGNOSI(**IDdiagnosi**, dataDiagnosi, CDC, *Idpaziente*);

e. MEDICO(**IDmedico**, CF, nome, cognome, via, numeroCivico, citta, provincia, regione, cap, dataNascita, sesso, email, password, idTelegram);

f. PAZIENTE(**IDpaziente**, CF, nome, cognome, stato, via, numeroCivico, citta, provincia, regione, cap, dataNascita, sesso, dataTrasferimento, email, password, cvindex, *IDmedico, Idstruttura*);

3. Implementazione fisica

L'implementazione fisica richiede troppe righe di codice per essere inserita qui, comunque si trova nel file **creazioneTabelle.php**.

CV-19 INDEX

CV-19 INDEX è un modello predittivo open source, scritto in python e basato su algoritmi di machine learning, sviluppato dal team di **ClosedLoop.ai**.

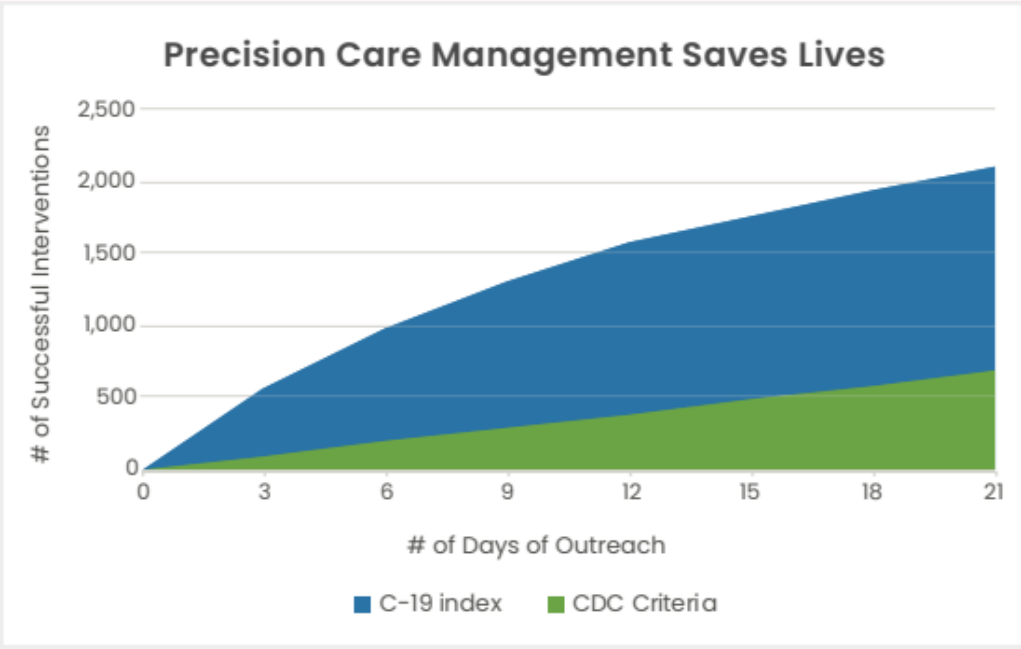
Chiunque sia interessato può scaricare il codice sorgente su **GitHub**, oppure semplicemente installlare la libreria via pip (per maggiori informazioni rimando alla pagina del progetto su **Pypi**). Inoltre, sul sito dell'organizzazione sopra citata è disponibile una versione hostata del programma che permette di calcolare personalmente il proprio indice di vulnerabilità.

Il programma pubblicato sfrutta il modello di machine learning "Gradient Boosted Trees", che combina insieme algoritimi più semplici in modo da ottenere una maggiore accuratezza nelle previsioni. Come dichiarato dagli stessi sviluppatori, algoritmi di questo genere hanno l'inconveniente di essere estremamente complessi e anche per loro sarebbe veramente arduo implementarli personalmente, quindi hanno utilizzato la libreria open-source XGBoost.

Siccome all'inizio del loro lavoro di ricerca non era disponibile una sufficiente quantità di dati relativi al Covid-19, per allenare e verificare il modello hanno utilizzato un dataset anonimizzato del programma di assicurazione medica statunitense Medicare, relativo agli anni 2015 e 2016, contenente le fatture mediche per il pagamento delle cure fornite. Dopo aver opportunamente selezionato i dati (relative ad un totale di 1,851,519 persone) e diviso il dataset in due parti (70% per il training set e 30% per il testing), hanno allenato l'algoritmo con i dati contenuti nelle cartelle cliniche dei pazienti, in modo da calcolare la probabilità di sviluppare severe complicazioni a partire

da infenzioni respiratorie simili al Covid-19 (per esempio polmonite e influenza), a seconda delle patologie preesistenti.

I risultati ottenuti sono soddisfacenti e lo strumento, abbinato alle linee guida fornite dall'OMS, può essere un aiuto prezioso ai medici di base nella cura tempestiva alla malattia (vedi grafico in figura tratto dal paper presente sul sito di ClosedLoop).



Siccome per ragioni di privacy non è possibile divulgare il dataset di riferimento, il programma condiviso contiene la rete precedentemente allenata e pronta all'uso.

Eseguire la libreria sui propri dati è molto semplice e i passaggi da seguire sono:

- Creare due file in formato CSV/Excel contenenti nell'ordine:
  - Id paziente, genere ed età dei pazienti;
  - Id paziente, id della fattura (io lo uso come identificativo della diagnosi), data del ricovero in ospedale o della visita, data di dimissione dall'ospedale (facoltativa), flag che indica se si tratta di una visita in pronto soccorso (0 falso, 1 vero), flag che indica se si tratta di un ricovero in ospedale (1 vero, altri valori falso), il codice ICD-10 della malattia/del sintomo/del disturbo in oggetto;

- Creare un file vuoto in formato CSV/Excel che conterrà le previsioni;
- Digitare su terminale cv19index file_pazienti file_diagnosi file_output;

A questo punto aprendo il file di output si troveranno diverse informazioni, ma l'unico preso in considerazione all'interno del sistema di monitoraggio è il risk_score. Tale valore rappresenta il percentile dell'indice di rischio rispetto alla popolazione americana (visto che il dataset è statunitense). Praticamente, un valore di 95 indica che l'indice di rischio è risultato maggiore rispetto al 95% della popolazione americana. Per avere maggiori informazioni sul software è possibile leggere il **paper** in allegato.

FUNZIONALITÀ DA IMPLEMENTARE

Oltre a quanto detto in precendenza a proposito della sicurezza, a causa del poco tempo a disposizione e della mancanza delle risorse necesssarie rimangono ancora da implementare diverse parti del sistema.

Innanzitutto occorre sviluppare un'applicazione cross-platform per smarphone, grazie alla quale ricevere i dati dal saturimetro e avviare lo script **application.py**. Per fare ciò si potrebbe pensare di sfruttare la libreria **Kivy**. Oltre a questo punto, occorre sviluppare tutto il sistema di registrazione al sistema, l'invio delle credenziali e delle comunicazioni via PEC e una procedura per il recupero e la modifica delle password.

Ovviamente oltre a quanto appena elencato sarebbe possibile ampliare il progetto a piacere, migliorando anche quanto già sviluppato, tuttavia non è strettamente necessario per il funzionamento del sistema.

ORGANIZZAZIONE DEL LAVORO

Per l’organizzazione del lavoro ho usato un metodologia che ricalca la filosofia delle metodologie agili. Infatti, non ho seguito una rigida scansione temporale, né delle delle fasi ben definite, bensì ho avuto un approccio orientato ai risultati (simile a quanto avviene con kanban), ritoccando spesso la struttura del database, modificando gli algoritmi e le librerie stesse.

Inoltre, mi sono sempre posto l'obiettivo di ottenere codice interamente funzionante ogni 2-3 ore di lavoro, in modo da non perdere tempo correggedo errori dovuti a delle implementazioni troppo complesse e avere sempre e comunque la certezza che tutto funzionasse come previsto.

Infine, durante lo sviluppo di ogni funzionalità ho adottato il ciclo di Deming, assicurandomi di ottenere una sufficiente qualità prima di proseguire.

Figura 10: Ciclo di Deming

SITOGRAFIA E RISORSE UTILI

Sito di ClosedLoop: https://closedloop.ai/

CV-19 index su GitHub: https://github.com/closedloop-ai/cv19index

CV-19 index su Pypi: https://pypi.org/project/cv19index/

Documentazione Paho: https://www.eclipse.org/paho/clients/python/docs/

Libreria telebot: https://github.com/eternnoir/pyTelegramBotAPI#general-api-documentation

Libreria Crypto: https://pycryptodome.readthedocs.io/en/latest/index.html

Libreria Kivy: https://kivy.org/#home

Libreria MySQL Connector: https://dev.mysql.com/doc/connector-python/en/

Libreria numpy: https://numpy.org/doc/stable/

Libreria pandas: https://pandas.pydata.org/docs/user_guide/index.html

Libreria XGBoost: https://xgboost.readthedocs.io/en/latest/build.html

Spiegazione algoritmo Gradient Boosted Trees: https://xgboost.readthedocs.io/en/latest/tutorials/model.html

Norma ISO/IEC 27001: https://www.iso.org/isoiec-27001-information-security.html

Normativa GDPR: https://www.garanteprivacy.it/web/guest/home/docweb/-/docweb-display/docweb/6264597

Informazioni utili sulla sicurezza nei database: https://www.alground.com/site/sicurezza-web-server-database/48795/

Front-end della dashboard: https://startbootstrap.com/themes/sb-admin-2/

Libreria chartjs per la creazione dei grafici: https://www.chartjs.org/docs/latest/

Prepared statement: https://www.php.net/manual/en/mysqli.quickstart.prepared-statements.php