Unito Pandemic

Riccardo Sieve*, Antonio Cavuoto*

*{riccardo.sieve, antonio.cavuoto}@edu.unito.it
Facoltà di Informatica, Università degli studi di Torino, Torino, Italia
Corso di Laurea Magistrale, corso di Programmazione per Dispositivi Mobili

25 Giugno 2020

Abbiamo portato avanti la progettazione e lo sviluppo dell'applicazione "Unito Pandemic" come parte del corso di Programmazione per Dispositivi Mobili con lo scopo di creare un'applicazione per Smartphone che soddisfasse determinati requisiti. Stante la situazione di emergenza affrontata nel 2020 a seguito dell'epidemia da SARS-CoV2, abbiamo deciso di portare avanti lo sviluppo di un'applicazione che potesse aiutare non solo nel periodo di crisi corrente, ma potesse fornire le basi per poter affrontare eventuali crisi future.

1 Introduzione

UniTO pandemic è un'applicazione molto semplice nata con lo scopo di aiutare la comunità scientifica a sviluppare modelli di predizione di un contagio virale, evitando la condivisione dei dati di geolocalizzazione degli utenti.

La modellizzazione della diffusione di malattie infettive è uno strumento matematico utile a studiare i meccanismi attraverso i quali si diffondono le malattie, in modo da fare predizioni e suggerire contromisure. Nel dettaglio, abbiamo diviso lo sviluppo dell'applicazione in tre parti distinte:

1. Applicazione mobile
2. Frontend
3. Backend

L'applicazione mobile è l'applicazione effettiva che gli utenti utilizzano e con cui si interfacciano; il backend è il server in cui transitano tutte le richieste di dati, mentre il frontend serve a fornire un'interfaccia per lo studio dei dati ottenuti per raffinare le eventuali euristiche fatte. L'intero sviluppo dell'applicazione è avvenuto seguendo i processi di security by design e privacy by design, andando pertanto a definire all'interno dell'intero processo di vita dello sviluppo dell'applicazione quali potessero essere eventuali problemi di sicurezza e/o privacy e andando a definire eventuali meccanismi e policy per evitare possibili exploit.

Per la condivisione dei dati tra utenti abbiamo adottato e implementato un protocollo di contact tracing decentralizzato che facesse utilizzo del bluetooth: il modello utilizzato per modellare questa struttura è il `DP-3T` [Prof Carmela Troncoso, 2020] che fa uso del bluetooth per generare dei beacon, nello specifico, vengono utilizzati due beacon distinti:

- Beacon per l'invio e la ricezione tramite bluetooth della propria chiave per il tracciamento dei contagi
- Beacon di simulazione per quanto riguarda l'invio e la ricezione delle chiavi per modellare la simulazione

L'applicazione ha lo scopo quindi di tenere sotto controllo i possibili contagi e, mediante futuri sviluppi, si prefigge l'idea di creare delle simulazioni per simulare possibili contagi futuri — per differenti possibili malattie — con probabilità di contagio differenti con lo scopo ultimo, da un lato consentire di poter intervenire in modo più tempestivo in caso scoppiasse una nuova pandemia, e dall'altro di fornire agli utenti una piattaforma che possa dare loro consigli su come comportarsi o come arginare nel miglior modo possibile i contagi.

Per fare questo, l'applicazione si basa su tre concetti:

1. **Performance** lato backend, utilizzando protocolli stateless mediante l'utilizzo di JWT[Sermersheim, 2015], e lasciando al client la parte computazionale (sopratutto per la generazione delle chiavi)
2. **Trasparenza** verso l'utente rendendo pubbliche le API utilizzate per leggere i dati all'interno del backend, di modo che gli utenti sappiano esattamente quali dati vengono trasmessi
3. **Privacy** by design e out of the box, mediante un'implementazione del protocollo `DP-3T` per la comunicazione

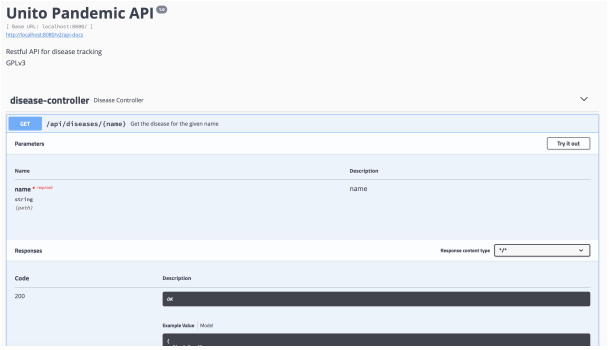

Figure 1: *Pagina di esposizione API*

2 Design dell'applicazione

2.1 Applicazione Android

L'applicazione Android è stata progetta per essere il più possibile responsive dal punto di vista della grafica, utilizzando strutture che si adattino al meglio a dispositivi differenti e che possa essere utilizzata sia con orientamento verticale o orizzontale del dispositivo.

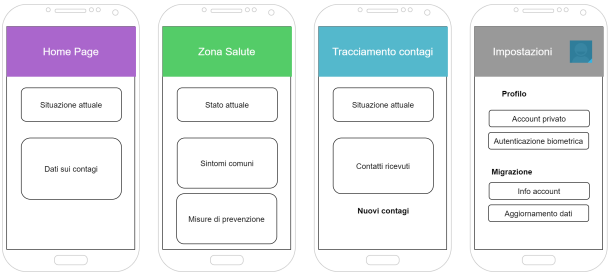

Figure 2: *Mockup dell'applicazione*

Nello specifico l'applicazione è stato strutturata in 4 macro componenti:

1. Il **Cruscotto** che funge da home page dell'applicazione; all'interno vi sono i dati presi dalla protezione civile mediante l'utilizzo di API REST che ci permettono di conoscere i contagi, e i dati da loro rilasciati
2. La **Zona Salute** contenente quelli che sono i sintomi comuni che possono essere consultati di modo da fornire una prima euristica su un possibile contagio, e le misure di prevenzione consigliate secondo le direttive dell'OMS
3. Il componente per il **Tracciamento dei contagi** contenente la lista di scambi di chiavi effettuati con altri dispositivi
4. Le **Impostazioni** che permettono ad un utente di impostare il proprio profilo come privato ed utilizzare eventualmente l'impronta biometrica

Ogni pagina utilizza un colore principale per il banner su cui è stato improntato lo stesso componente.

Tutta questa parte è predisposta al fatto che l'utente sia già registrato; qualora non fosse così dovrà prima registrarsi fornendo:

- Nome (o nickname a discrezione dell'utente) con cui verrà registrato
- La propria età e sesso — utilizzate per effettuare analisi statistiche

mentre all'accesso dell'applicazione verranno mostrate all'utente delle semplici schermate per fornire all'utente il contesto su cui orbita l'applicazione.

2.2 Backend

Il backend su cui vengono fatte le chiamate per inserire e/o ottenere i dati si basa sulla tecnologia `Spring`[1], sulla base di `Java 11`[2], utilizzando `PostgreSQL`[3] come database

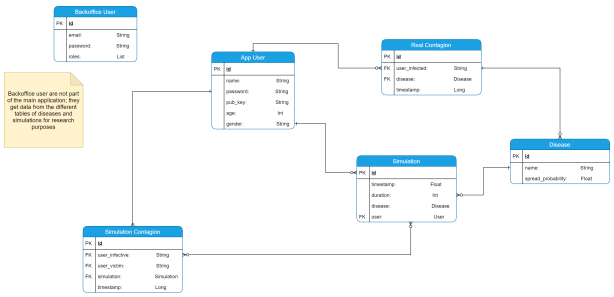

Figure 3: *Schema del db*

La struttura del database è stata progettata mediante `JPA`[4] per la persistence, definendo le varie classi che compongono le tabelle del database. Per la modellazione dei dati, il backend è stato strutturato secondo un sistema architetturale basato su microservizi, seguendo le linee guida del pattern MVC[5].

[1] `https://spring.io/why-spring`
[2] `https://blog.idrsolutions.com/2019/05/java-8-vs-java-11-what-are-the-key-changes/`
[3] `https://www.postgresql.org/about/`
[4] `https://www.oracle.com/technical-resources/articles/java/jpa.html`
[5] `https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#cite_note-1`

2.3 Frontend

Il frontend serve come piattaforma per lo studio dei dati ottenuti dall'utilizzo dell'applicazione mobile, in quanto tale la struttura è limitata ai semplici componenti che servono a questo scopo:

- La sezione riguardanti le malattie per poter definire con più precisione gli eventuali tassi di contagio
- La sezione — accessibile ai soli amministratori — per definire o cambiare i ruoli degli utenti della piattaforma
- La sezione per la visualizzazione dei dati delle malattie

L'idea, alla base di questa piattaforma, è di consentire, in un'implementazione futura, di collaborare con matematici e biologi per poter meglio tracciare le possibili viralità degli agenti pandemici di modo da isolare e meglio combatterli con maggiore tempestività.

3 Implementazione

3.1 Applicazione Android

Abbiamo sviluppato l'applicazione utilizzando Flutter[6] con il linguaggio di programmazione `Dart` per sviluppare un'applicazione che fosse moderna, veloce, e cross-platform, in questo modo è possibile fin da subito impostare l'intera applicazione perché possa essere utilizzata, con il minor numero possibile di modifiche, su piattaforme Android e iOS.

A seguito della registrazione, l'applicazione genera tre chiavi distinte seguendo lo standard delle curve di Edwards **ed25519** [D.J. Bernstein, 2011] come segue:

1. Una chiave per la comunicazione con il backend: il server mappa le chiavi corrispondenti per gli utenti per capire quale utente sta effettuando le chiamate
2. Una chiave per la comunicazione con gli altri dispositivi per la mappatura dei contatti durante la normale esecuzione dell'applicazione
3. Una chiave per la comunicazione con gli altri dispositivi **fatte durante la simulazione**

La distinzione tra le ultime chiavi serve per comprendere quando un contatto avviene durante un contagio (e permette quindi di avere euristiche sui contagi legate allo studio di possibili contagi per malattie simulate) simulato, mentre l'altra chiave mappa i contatti che possono portare ad un contagio effettivo.

Lo scambio delle chiavi viene fatto mediante beacon bluetooth utilizzando un'implementazione del protocollo `DP-3T`

[6] `https://flutter.dev/`

`DP-3T` Il `Decentrilized Privacy Preserving Proximity Tracing` è un protocollo atto per fornire una comunicazione decentralizzata, sicura, e privacy orienting per permettere la comunicazione tra dispositivi tramite protocollo bluetooth low energy. L'idea che sta alla base di questo protocollo è piuttosto semplice: utilizziamo **un beacon bidirezionale**: una direzione per la ricezione dei dati e una per l'invio.
I dispositivi mettono a disposizione i propri dati mediante `advertisement` del beacon di invio; non appena un altro dispositivo entra in contatto via bluetooth, verrà segnato all'interno del componente atto al tracciamento dei contagi.

Attualmente vengono salvate le sole chiavi con cui entriamo in contatto; questo viene fatto per un semplice motivo: a seguito di un effettivo contagio mostrato da un medico, quest'ultimo può segnare la chiave del malato all'interno della lista resa pubblica, in questo modo tutti gli utenti possono sapere se le persone con cui sono venuti a contatto sono o meno contagiati — questo può fornire delle euristiche per prendere eventuali precauzioni, controlli con un medico per valutare un eventuale contagio e così via.

Va fatto notare che questo utilizzo del protocollo è privacy preserving dal momento che l'inserimento di un utente all'interno della lista dei contagiati non mostra alcun tipo di informazione riguardo all'utente, se non la propria chiave (quest'ultima è conosciuta solo dall'utente e non viene salvata nel server centrale, rendendo quindi gli utenti sicuri da possibili `identity disclosure`).

Tutte le informazioni riguardo lo scambio di chiavi tra utenti vengono salvate solo localmente mediante l'utilizzo di un database locale (cifrato in `AES-256`) senza mandare alcun tipo di informazione al server centrale di modo da assicurare la sicurezza e la segretezza dei dati degli utenti.

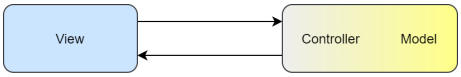

Figure 4: *Pattern MVC dell'applicazione*

Per quanto riguarda la struttura del codice, abbiamo seguito il pattern `MVC` la cui struttura si discosta leggermente da quella canonica, in quanto `Controller` e `Model` vengono pressoché accorpati, mentre la distinzione avviene tra questo blocco e la `View`.

La scomposizione del codice ci permette di avere una maggiore modularità e controllo sulla struttura dell'applicazione e ci permette di avere una maggiore ed efficiente manutenzione, così come la possibilità di ampliamento e scalabilità.

In ultima analisi, per la navigazione all'interno

dell'applicazione si è fatto uso delle gesture, di modo da consentire un utilizzo più fluido e moderno, migliorando la user experience con l'app.

3.2 Backend

Per il backend, abbiamo utilizzato un approccio a microservizi con persistence `JPA`

- Il `model` è la struttura delle classi che va a modellare le varie tabelle del database e che vengono passati poi all'applicazione e al frontend come oggetti
- Il `controller` funge da intermediario per le chiamate REST per invocare i vari servizi; riceve le chiamate ed opera mediante i `repository` per il passaggio dei dati del database

Figure 5: *Schema di JWT*

Il passaggio di tutte le chiamate e dei dati avviene mediante `REST API` e utilizza il protocollo di `JWT` per permettere un'autenticazione stateless e scalabile.

`JWT` Il `JSON Web Token` comprende tre componenti principali:

1. Un **Header** contenente l'algoritmo e il tipo di token
2. Il **Payload** contenente il dato effettivo passato
3. La **Signature** che contiene la firma per la verifica, effettuata come `HMAC` con `SHA 256`

3.3 Frontend

La dashboard utilizza `Vue.js`[7] con `node`. Si basa sull'utilizzo di components definiti dirrettamente come snippet di codice `html`, chiamati `template`, che vengono poi richiamati all'interno delle varie pagine.

Le chiamate vengono fatte indipendentemente dagli utenti che vi accedono, con la differenza che i ruoli indicheranno quali pagine potranno essere visitate e quali dati saranno visualizzati:

[7] https://vuejs.org/

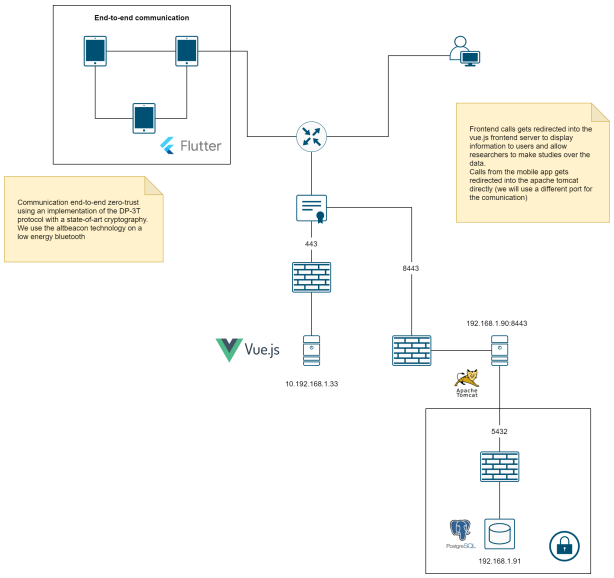

Figure 6: *Progettazione del flusso applicativo*

- Gli utenti base (che non hanno ancora ottenuto un ruolo ben definito) hanno semplicemente accesso alla dashboard senza poter fare modifiche (accesso in semplice lettura alla dashboard e alla visualizzazione dei dati)
- I ricercatori possono visualizzare nel dettaglio i dati degli utenti (nello specifico le chiavi dei contagiati) ed eventualmente modificare i parametri di probabilità di contagio
- I dottori potranno, a seguito di una visita con un paziente (e la notifica dello stesso della propria chiave) segnalare la chiave come chiave di un paziente contagiato, di modo che gli utenti entrati in contatto con esso possono vedere la chiave ed eventualmente procedere con ulteriori controlli
- Gli amministratori possono accedere ai dati e modificare i ruoli degli utenti appartenenti alla piattaforma

4 Conlcusioni e sviluppi futuri

Lo sviluppo dell'applicazione ha portato alla creazione di un **prototipo verticale** che permette di mostrare, sotto forma di demo, le funzionalità principali dell'applicazione, sebbene quest'ultima non sia ancora *production ready*.

Nello stato attuale, comunque, l'applicazione fornisce una maggiore trasparenza per l'utente, andando a mostrare chiaramente tutte le informazioni che vengono passate, così come la struttura delle API, come forma di fidelizzazione per gli utenti che decidono di adottarla; in particolare, viene assicurata la privacy, evitando da un lato di richiedere dati sensibili ai fruitori, dall'altro utilizzando la tecnologia bluetooth low energy con un'implementazione custom del protocollo `DP-3T` consentendo il tracciamento dei

contagi senza dover ricorrere alla geolocalizzazione.

Abbiamo fatto degli studi riguardo all'interfaccia utente e alla user experience per assicurare una comoda fruibilità dell'applicazione da un gruppo più grande possibile di utenti, mediante layout fluidi, veloci e moderni; l'utilizzo di Flutter e del linguaggio di programmazione `Dart` ha portato allo sviluppo di un'applicazione **cross-platform** evitando di utilizzare elementi che richiedevano lo sviluppo di codice nativo per accedere a funzionalità che non sarebbero accessibili altrimenti.

4.1 Sviluppi futuri

Sebbene sia stato implementato un prototipo verticale funzionante abbiamo dovuto fare un compromesso a ribasso sulle varie funzionalità da includere a causa della defezione dal gruppo dello studente Tagliaferro. Inizialmente, infatti, il progetto era stato pianificato per essere portato avanti da tre persone con una tempistica di base di un mese e mezzo.

Nello specifico, le funzionalità che vorremmo aggiungere all'applicazione sono:

- Utilizzo della variabilie per la probabilità di contagio: sebbene la variabile sia presente sia nel codice che nel database, attualmente non viene sfruttata; per motivi di tempo, in questo prototipo abbiamo preferito mappare semplicemente i contatti tra utenti, e lasciare al medico lo scopo di segnare le chiavi degli utenti che hanno effettivamente contratto il virus, per essere poi pubblicate nella lista di contagi.
  Un possibile metodo di analisi direttamente machine-to-machine può essere rappresentato dal calcolo della suscettibilità ad un patogeno mediante un sistema di equazioni differenziali tipiche del modello SIR[8] come segue:

$$\frac{ds}{dt} = -bs(t) \cdot i(t) \tag{1}$$

- A seguito della registrazione, l'applicazione genera tre chiavi distinte seguendo lo standard delle curve di Edwards **ed25519** [D.J. Bernstein, 2011]
- Collegamento per consentire una lista di news feed personalizzato
- Aggiungere un collegamento con il Sistema Sanitario Regionale, questo per consentire un monitoraggio attivo da parte del medico (ad esempio consentendo di effettuare visite via videochat)
- Abilitazione, **solo in locale**, della geolocalizzazione per il tracciamento di propri spostamenti mediante l'utilizzo di `OpenStreetMaps`
- Passaggio all'utilizzo di API di esposizione di Google e Apple al posto della nostra soluzione Nearby che, per quanto funzionante, ha precluse alcune funzionalità, quali il background e la velocità

[8] `https://www.maa.org/press/periodicals/loci/joma/the-sir-model-for-spread-of-disease-the-differential-equation-model`

Riferimenti

[D.J. Bernstein, 2011] [1] D.J. Bernstein, N. Duif, T. L. P. S. B.-Y. Y. (2011). "High-speed high-security signatures". Technical report nº. `https://ed25519.cr.yp.to/ed25519-20110926.pdf`.

[Prof Carmela Troncoso, 2020] [2] Prof Carmela Troncoso, . (2020). "Decentralized Privacy-Preserving Proximity Tracing". Technical report nº. `https://github.com/DP-3T/documents/blob/master/DP3T%20White%20Paper.pdf`.

[Sermersheim, 2015] [3] Sermersheim, J. (2015). "Json web token (jwt)". RFC 7519, *Internet Engineering Task Force*. `https://tools.ietf.org/rfc/rfc7519.txt`.

[Weiss, 2013] [4] Weiss, H. (2013). "The SIR modeland the-Foundations of Public Health". Technical report Nº 3, *MATerials MATemàtics*. `http://mat.uab.cat/matmat_antiga/PDFv2013/v2013n03.pdf`.