UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Ingegneria “Enzo Ferrari”

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

Sviluppo di funzionalità in D3.js per applicazione di data-driven visualization volta al monitoraggio degli spostamenti di pazienti COVID all’interno di un ospedale.

Relatore

Sonia Bergamaschi

Correlatore

Giovanni Simonini

Candidato

Alessio Pugnaghi

SOMMARIO

In questo elaborato viene presentata e descritta l'attività progettuale svolta con i correlatori Giovanni Simonini e Luca Zecchini, per l'azienda di Reggio Emilia *Coopservice S. Coop. P.A*. Quest'ultima infatti ha commissionato all'università un progetto riguardante lo sviluppo e la realizzazione di un'applicazione web per lo spostamento dei pazienti dell'ospedale Malpighi-Sant'Orsola di Bologna.

Si è rivolta particolare attenzione allo spostamento dei pazienti affetti da Covid, in modo tale da ottimizzare i percorsi e le risorse, sia umane (i trasportatori) che non, durante l'arco di intere giornate. Il periodo che inizialmente si è preso in considerazione va da marzo ad agosto del 2020, ovvero l'inizio della pandemia, con i dati che ci sono stati forniti direttamente dall'azienda sottoforma di tabelle Excel.

Come vedremo in seguito per la realizzazione dell'applicazione, il cui obiettivo è appunto quello di poter visualizzare in maniera piuttosto precisa e corretta i diversi spostamenti giornalieri, di cui si terrà traccia anche attraverso appositi grafici e contatori, sono stati utilizzati diversi tool informatici e software, come ad esempio Visual Studio Code, Eclipse e altri ambienti di programmazione; mentre i linguaggi utilizzati sono stati perlopiù HTML e JavaScript con l'aiuto della libreria D3, volta proprio a migliorare la visualizzazione grafica dell'output del codice.

L'output mostrato sarà dunque una visualizzazione dinamica e schematica dei vari ambienti dell'ospedale suddivisi per Padiglioni, Piani o Reparti. Gli spostamenti invece saranno riconoscibili attraverso piccoli cerchi colorati che si muoveranno tra i vari ambienti e assumeranno colori diversi a seconda della rappresentazione che si vuole visualizzare. Si terrà traccia inoltre anche del lavoro svolto dai trasportatori, entità che hanno assunto un ruolo fondamentale nel corso della pandemia.

All'interno di questo elaborato saranno presenti alcune immagini con lo scopo di rendere l'idea di ciò che è stato ottenuto insieme all'azienda.

INDICE

CAPITOLO 1-Introduzione

Con il presente elaborato, descriverò la mia esperienza riguardante l'attività progettuale svolta per l'azienda *coopservice* insieme al collega Gabriele Ruini e ai correlatori Giovanni Simonini e Luca Zecchini.

In particolare tratterò del mio contributo personale, avvenuto a progetto già iniziato dal mio collega e portato avanti fino ad ora.

L'oggetto di questa relazione sarà quindi la descrizione del processo, dalle prime fasi di studio fino all'implementazione di un prototipo di applicazione web funzionante, per la ottimizzazione degli spostamenti dei pazienti affetti da Covid all'interno dell'ospedale Sant'Orsola.

Guardando i vari capitoli, nel secondo si andrà ad analizzare il contesto generale, spiegando quali sono gli obiettivi voluti dall'azienda e le motivazioni della stessa.

Nel terzo capitolo saranno introdotte e descritte le tecnologie informatiche e i software utilizzati per la realizzazione del progetto, ponendo un'attenzione particolare nel quarto capitolo a D3.js, una libreria di JavaScript volta a migliorare le rappresentazioni dinamiche dei dati.

Per il quinto capitolo mi concentrerò sulla presentazione del lavoro da me svolto, con immagini e screenshot dell'applicazione.

Nel sesto capitolo si andrà a illustrare le possibilità future per l'applicazione e le varie opportunità ottenibili continuandone lo sviluppo.

CAPITOLO 2- Contesto generale

In questo secondo capitolo verranno analizzati tutti gli aspetti riguardanti l'azienda e l'assegnazione del progetto.

Nel primo paragrafo in particolare ci sarà una descrizione generale di Coopservice, partendo da un piccolo excursus sulla sua storia, dalla fondazione fino a oggi, e dell'ambito lavorativo in cui opera, con alcuni servizi da essa erogati.

Nel secondo parametro si passerà a un'analisi più approfondita riguardo il progetto su cui è stato svolto questo elaborato, con particolare attenzione verso i dati che ci sono stati forniti riguardo l'ospedale Sant'Orsola di Bologna.

Il terzo paragrafo invece è dedicato agli obiettivi che ci siamo posti per la realizzazione e lo sviluppo dell'applicazione.

2.1 Azienda Coopservice

Coopservice è un'azienda, discretamente conosciuta sul territorio emiliano, che opera in diversi ambiti, come civile, sanitario e aziendale, erogando servizi di facility management, tra cui:

- Pulizia e igiene
- Privacy e vigilanza
- Movimentazione merci e logistica

Per tutti quesiti servizi viene garantito un elevato livello di sicurezza, efficienza e comfort.

Coopservice nasce nel 1991 dalla fusione di due importanti aziende di servizi attive nella vigilanza e nella pulizia, Coopsicurezza (nata nel 1976) e Cierrepi(1972). La prima sede è in via Galliano a Reggio Emilia.

Nel corso degli anni acquisisce diverse società e aziende italiane espandendosi sempre di più nel resto della penisola. Nel 2009, vista la continua crescita, c'è la necessità di cambiare sede e perciò si trasferisce da Cavriago a Pratofontana di Reggio Emilia, in Via Rochdale n. 5. A poche centinaia di metri sta nascendo la stazione Medio Padana sulla linea ad Alta Velocità Milano-Roma.

Ad oggi, nel 2021, 30 anni dopo la fondazione dell'azienda si è arrivato, dopo un'enorme espansione che ha valicato persino i confini italiani, ad avere 25000 dipendenti e un fatturato che tocca il miliardo di euro.

Per l'ambito sanitario Coopservice ha sempre erogato servizi volti a soddisfare un sistema di gestione che garantisca la qualità in tutta la sua totalità, detto TQM (Total Quality Management, ovvero una tecnica di gestione della qualità che, essenzialmente, si basa sul fatto che tutti i membri di un'organizzazione puntino a migliorare continuamente la loro svolgere le loro funzioni, nell'ottica dell'obiettivo finale della soddisfazione del cliente).

A partire da marzo 2020 in tutta Italia i servizi sanitari sono diventati fondamentali per poter contrastare l'esplosione della pandemia e per poter garantire le attività essenziali negli ospedali. Questo impegno è stato preso in prima linea dall'azienda che ha offerto con continuità servizi di igiene e sanificazione, logistica, vigilanza e sicurezza negli ospedali, nei trasporti, nei centri commerciali, nei luoghi di lavoro per prevenire e contrastare la diffusione del virus.

Questi servizi sono inoltre erogati secondo i valori e la mission che Coopservice si è posta negli anni per essere leader nel mondo, con qualità e innovazione del lavoro, mettendo al centro i clienti e la collettività, valorizzando la professionalità e le aspirazioni dei propri soci e lavoratori.

I valori in cui si riconosce l'azienda sono:

- Lavoro di squadra
- Coerenza
- Trasparenza
- Innovazione
- Legalità

Grazie a questi principi e alle attività essenziali garantite, gli ospedali che hanno collaborato con Coopservice, hanno potuto mantenere ambienti con elevati livelli di igiene e pulizia, caratteristiche davvero fondamentali visto il sovraccarico affrontato dalle strutture ospedaliere. In questo modo si è potuto salvaguardare la salute dei pazienti, degli ospiti e degli operatori (medici, trasportatori, collaboratori, ecc..), riuscendo anche ad intervenire tempestivamente in caso di danno alla sicurezza, vista la continua ricerca di ottimizzazione delle risorse e della logistica. Quest'ultimo fatto è visibile all'interno degli edifici, locali e impianti dove Coopservice è intervenuta andando a raggiungere un alto livello di efficienza, con il minimo spreco di risorse e trasporti.

La logistica e la sua gestione sono aspetti molto importanti per l'azienda e riguardano una buona parte dei suoi progetti, soprattutto all'interno di strutture ospedaliere, dove con l'avvento della pandemia e la messa in quarantena di interi reparti, ci si è affidato maggiormente ad enti esterni per gestire e ottimizzare gli spostamenti di pazienti e collaboratori, in modo tale da potersi concentrare esclusivamente sulla cura del paziente, la loro missione primaria.

Gli enti a cui ci si affida sono altamente specializzati e insieme all'ospedale dovranno condividere gli obiettivi per raggiungere dei risultati prefissati. Inoltre, sgravandosi delle funzioni logistiche, le aziende ospedaliere possono ridurre i costi (sia monetari che di risorse) e assicurare una maggiore flessibilità.

Così facendo si potranno creare delle condizioni per cui le aziende sanitarie, insieme a questi enti di gestione della logistica, possano migliorare tutti i loro servizi come la distribuzione di prodotti farmaceutici, dispositivi medico-chirurgici e prodotti economali, e anche le attività logistiche al proprio interno (trasporto dei pazienti, magazzini, archivi, distribuzione di cibo e medicinali, turni di lavoro).

Riassumendo gli obiettivi che Coopservice vuole perseguire nella logistica ospedaliera sono:

- Rilevare le funzioni logistiche, lasciando all'azienda ospedaliera il compito di concentrarsi solamente sulla propria mission di cura del paziente.
- Ottimizzare, incrementarne la flessibilità e ridurre i costi per le attività logistiche dell'azienda ospedaliera con cui si collabora.

Tra le aziende ospedaliere che si sono affidate a Coopservice c'è l'ospedale Sant'Orsola di Bologna, accordo che è alla base di questo progetto e di questo elaborato.

Durante i primi mesi della pandemia, in particolare dai primi giorni di marzo 2020, l'azienda reggiana ha tenuto conto degli spostamenti che sono avvenuti all'interno dell'ospedale, tra diversi reparti, padiglioni o piani, con particolare attenzione rivolta verso i pazienti affetti da Covid-19, per i quali bisognava evitare contatti con gli altri pazienti negativi. Come vedremo meglio nel prossimo paragrafo i dati (decine di migliaia) sono stati raccolti sotto forma di tabelle Excel, salvando diverse informazioni che saranno utili nello sviluppo dell'applicazione.

L'analisi di quest'ultimi è oggetto della collaborazione tra la nostra università, Unimore, e Coopservice, in modo da poter visualizzare dinamicamente e graficamente queste tabelle, altrimenti difficili da leggere, attraverso lo sviluppo e la realizzazione di un'applicazione web.

2.2 Le tabelle Excel e i dati

Le tabelle che ci sono stati fornite da Coopservice sono state realizzate in base a dati reali, dalla loro Area di Ricerca e sviluppo. Essi forniscono una grande base di dati, con informazioni relativi ai piani, padiglioni e reparti di partenza e arrivo relativi a uno spostamento. Facilmente intuibile il costo diverso (in termini di risorsa umana, ovvero i trasportatori, che possono essere uno oppure due) in base alla lunghezza e alla durata di uno spostamento. In particolare vengono analizzati quelli tra piani differenti dell'ospedale, ritenuti i più dispendiosi dal punto di vista di energia umana utilizzata.

Tra le altre informazioni che abbiamo ci sono quelle relative alle modalità di trasporto e della positività o meno di un paziente, casi molto importanti vista l'emergenza sanitaria e con costo maggiore rispetto a spostamenti normali, con l'obbligo di prendere maggiori precauzioni.

I dati raccolti sono strutturati in 3 diversi file Excel, sotto forma di tabelle, dove ogni riga rappresenta un singolo spostamento e ogni colonna conterrà un'informazione particolare relativa a quello spostamento (le analizzeremo attentamente in seguito).

Il primo file contiene gli spostamenti avvenuti tra l'1 marzo 2020 e l'1 agosto 2020 (più di 30mila), il secondo raccoglie un periodo che va da 2 agosto 2020 e 31 dicembre 2020 (anche questi più di 30mila), mentre l'ultimo, il più piccolo e il più recente, quelli relativi a gennaio 2021 (più di 6mila).

Il totale degli spostamenti di cui si è tenuto traccia ammonta a 70.391.

Come anticipato, per ogni spostamento si è tenuto traccia di diverse informazioni, suddivise per colonne, di cui le più importanti:

- **Data/Ora Inserimento,** relativo all'inserimento dello spostamento all'interno del file.
- **Nominativo Paziente,** stringa non completa del nome del paziente per motivi di privacy, ma che riporta l'iniziale del nome e le prime due lettere del cognome
- **Modalità Trasporto,** sono 7: letto, carrozzina normale, deambulatore, barella obesi, carrozzina obesi e giro a vuoto.
- **Priorità,** stringa che indica la motivazione dello spostamento e può essere: trasferimento, urgenza clinica, rientro, emergenza, urgenza organizzativa, non programmato, programmato, trasferimento con percorso programmato.
- **Tipologia Trasporto,** seconda stringa che dà informazioni sulla motivazione di uno spostamento e può essere: trasferimento, esame strumentale, rientro, consulenza, multi-consulenza oppure altro.
- **Reparto / Unità Operativa Partenza.**
- **Reparto / Unità Operativa Arrivo.**
- **Trasportatore 1,** stringa che indica il cognome dell'operatore sanitario ("so.cognome"), che effettua il trasporto.
- **Trasportatore 2,** stringa che indica l'eventuale cognome di un secondo operatore sanitario.
- **Data Inizio,** indica la data e l'ora dell'inizio delle operazioni di trasporto.
- **Data Fine**, indica la data e l'ora della conclusione delle operazioni di trasporto.
- **O / V,** indica se lo spostamento effettuato è orizzontale (stesso piano) oppure verticale (piani differenti).

Oltre a queste ci sono ulteriori informazioni che sono registrate, ma che al fine del progetto non sono risultate essenziali, ma comunque sono state salvate:

- **Note** varie, riferite ad aspetti atipici del paziente, dell'ambiente circostante o del trasporto
- **Note Centrale Operativa,** ovvero che arrivano direttamente dalla centrale operativa.
- **Data Ricezione,** data e ora dell'avvenuta chiamata dalla centrale operativa verso un operatore.
- **Presa in Carico,** data e ora della presa in carico di uno spostamento da parte di un operatore.
- **Fine OP Mobile,** coincide con la suddetta colonna Data Inizio, quindi non aggiunge informazioni.
- **Ingresso Rep P,** data e ora dell'ingresso di un operatore nel reparto da cui parte uno spostamento.
- **Uscita Rep P,** data e ora dell'uscita di un operatore nel reparto da cui parte uno spostamento.
- **Ingresso Rep A,** data e ora dell'ingresso di un operatore nel reparto in cui arriva uno spostamento.

- **Uscita Rep A,** data e ora dell'uscita di un operatore nel reparto in cui arriva uno spostamento.
- **Chiusura DEF,** indica la chiusura definitiva delle operazioni di trasporto.

Oltre a questi 3 file ci è stato fornito un'ulteriore tabella, che aveva la funzione di elencare tutti i reparti dell'ospedale Sant'Orsola suddivisa per padiglioni e piani, per un totale di circa 300 unità. Come per i file precedenti i reparti sono rappresentati tramite righe, mentre le informazioni ad essi relative sono state suddivise per colonne:

- **Padiglione**, indica il padiglione in cui il reparto è situato.
- **Piano**, indica il piano in cui il reparto è situato.
- **Reparto / Unità Operativa**, nome completo del reparto, molto utile per fare match con i primi 3 file.
- **Numero di telefono**, stringa di numeri non utilizzata per lo scopo del progetto.
- **Short**, piccola descrizione del reparto che ho aggiunto per raggrupparne alcuni poiché altrimenti sarebbe stato impossibile rappresentarli tutti. Nella rappresentazione finale sarà infatti visibile questa e non il nome intero.

2.3 Obiettivo del progetto

Ciò che Coopservice ci ha chiesto di realizzare, è di poter visualizzare tutti gli spostamenti raccolti nei file in modo dinamico e compatto.

L'idea quindi iniziale era quella di poter selezionare una data (o diverse date) e vedere graficamente su una planimetria apposita, tutti gli spostamenti di quel periodo.

Per realizzare questa prima richiesta, si è deciso di rappresentare ogni singolo spostamento come un piccolo cerchio, e quindi il risultato visibile era quello di tanti "pallini" che si muovevano da un padiglione all'altro oppure da un piano all'altro all'interno dell'ospedale Sant'Orsola-Malpighi, seguendo le date e le ore indicate nei file.

A seguito di un incontro diretto con l'azienda si è pensato di tenere anche traccia di diverse metriche, che riguardassero in particolar modo gli operatori, come il tempo medio che lavorano in un proprio turno o il numero di spostamenti eseguiti. Inoltre questo sarà visibile anche attraverso le diverse colorazioni dei pallini, che andranno ad indicare chi è l'operatore che ha in carico lo spostamento.

Come già detto in precedenza, essendo il numero di reparti molto elevato, si è implementata la visione degli spostamenti solamente tra piani e padiglioni al momento, ovvero gli spostamenti più dispendiosi. Si potrà comunque accedere ai reparti di uno specifico padiglione e/o piano. In questo caso comunque non apparirà il nome completo, ma una descrizione che è stata da me introdotta direttamente nel file per una migliore rappresentazione durante la navigazione.

CAPITOLO 3- Tecnologie Utilizzate

Nel corso di questo capitolo presenterò le varie tecnologie, software, linguaggi di programmazione e ambienti di sviluppo utilizzati. In questo modo risulterà più chiaro al lettore ciò che è stato presentato in precedenza e ciò che verrà presentato in seguito nel capitolo relativo al progetto.

Nei primi due paragrafi saranno descritti i due ambienti di sviluppo che ho utilizzato, ovvero *Eclipse e Visual Studio Code.*

Nel terzo ho inserito un piccolo excursus su *Excel* e le sue tabelle, con indicate le operazioni che ho svolto per modificare i file consegnati dall'azienda.

Nei restanti 3 capitoli invece vengono descritti i linguaggi di programmazione che sono stati utilizzati per lo sviluppo e la realizzazione di questo progetto, linguaggi fondamentali nella programmazione web: *CSS, HTML e JavaScript.*

3.1 Eclipse

Eclipse è un ambiente di sviluppo integrato multi-linguaggio e multipiattaforma. Ideato da un consorzio di grandi società, è un software libero distribuito sotto i termini della Eclipse Public License. La prima versione fu rilasciata il 7 novembre 2001 e da allora è stato aggiornato diverse volte fino ad arrivare all'ultima (4.21) del settembre 2021.

Eclipse può essere utilizzato per la produzione di software di vario genere, si passa infatti da un completo IDE per il linguaggio Java (JDT, "Java Development Tools") a un ambiente di sviluppo per il linguaggio C++ (CDT, "C/C++ Development Tools") e a plug-in che permettono di gestire XML, JavaScript, PHP e persino di progettare graficamente una GUI per un'applicazione JAVA (Window Builder). Naturalmente supporta anche linguaggi per la creazione di servizi web come Html e CSS.

Offre diversi tools utili alla programmazione, già incorporati al suo interno, come il debugging, la creazione di Servlet e progetti Web Dinamici, fondamentali per lo scopo di questo progetto.

Con questo ambiente avevo già familiarità, avendolo usato in diversi esami durante l'esperienza universitaria e perciò l'ho preferito a Visual Studio Code, che invece ho utilizzato inizialmente.

3.2 Visual Studio Code

Visual Studio Code è un editor di codice sorgente sviluppato da Microsoft per Windows, Linux e macOS. È un software libero e gratuito, anche se la versione ufficiale è sotto una licenza proprietaria. La prima versione è stata rilasciata da Microsoft nel novembre 2015, mentre l'ultima risale all'ottobre 2021. Si tratta quindi di un editor abbastanza giovane e perciò decisamente innovativo e all'avanguardia

Visual Studio Code è un editor che può essere usato con vari linguaggi di programmazione, tra cui la famiglia di linguaggi C (C, C++, C#), Java, HTML, CSS, JavaScript, Phyton e altri linguaggi web.

Come Eclipse mette subito a disposizione molte funzionalità come il debugging (grazie a un supporto integtato per JavaScript), oltre ad altri debugger facilmente installabili dal *market-place.* Da quest'ultimo si possono anche scaricare tante estensioni a seconda del codice che si deve sviluppare. Visual Studio Code infatti può essere molto utile anche grazie alla sua grande flessibilità.

Altri servizi aggiuntivi possono essere *syntax highlighting* (o colorazione della sintassi), di *IntelliSense* e di *refactoring* del codice.

3.3 Excel e le sue tabelle

Questo paragrafo ha la funzione di spiegare meglio il lavoro da me svolto sulle tabelle Excel che contenevano i dati raccolti dall'azienda, lavoro che ho in parte già spiegato nel capitolo precedente.

Microsoft Excel è un programma prodotto da Microsoft, dedicato alla produzione ed alla gestione di fogli elettronici. È parte della suite di software di produttività personale Microsoft Office, ed è disponibile per i sistemi operativi Windows e macOS. È il programma per la produzione e gestione di fogli elettronici più utilizzato.

La prima versione è stata rilasciata nel 1985 per il SO Macintosh. L'ultima invece risale a ottobre 2021.

Lo scopo per cui viene più usato questo software probabilmente è quello di creare tabelle, dinamiche o statiche, che fungono da base di dati, talvolta, come nel nostro caso, anche per applicazioni web. Infatti negli anni Excel ha aggiunto diverse funzionalità volte a semplificare e velocizzare la modifica di queste tabelle, come ho potuto sperimentare in prima persona, andando ad aggiungere centinaia di dati, per poter rendere migliore la visualizzazione finale del progetto.

3.4 HTML

In informatica l'HyperText Markup Language, comunemente noto con l'acronimo HTML, è un linguaggio di markup ipertestuale. Venne sviluppato nei primi anni novanta da Tim Berners-Lee (co-inventore del World Wide Web) al CERN di Ginevra, in Svizzera, assieme al protocollo HTTP dedicato al trasferimento di documenti in tale formato.

L'ultima versione rilasciata è HTML5 del 2014, sviluppata dal World Wide Web Consortium, che ha aggiunto ulteriori funzionalità per stare al passo con il boom di internet e degli smartphone.

HTML è un linguaggio di markup gerarchico strutturato ad albero: esistono collegamenti gerarchici fra gli elementi, che rendono uno l'antenato (o genitore) dell'altro (discendente o figlio).

Il linguaggio HTML consente di descrivere semanticamente la struttura di un documento web attraverso tag, ad esempio identificando una sezione di testo come intestazione, paragrafo, elenco, collegamento, citazione o altro elemento. HTML rappresenta dunque la struttura portante

delle pagine web: su questa struttura si possono aggiungere modifiche grafiche, grazie ai fogli di stile CSS, ed elementi dinamici, grazie alla programmazione JavaScript. HTML, CSS e JavaScript sono le tre tecnologie alla base della programmazione front end, come vedremo anche in seguito.

Alla base, quindi, di HTML ci sono i cosiddetti **<tag>**, keyword che corrispondono a determinati tipi di contenuto. Ogni tag può avere degli attributi specifici, cosa che permette di costruire pagine diverse fra di loro e che rispondano alle necessità di chi le scrive. Così facendo avremo in output delle strutture ad albero, che rispettano una gerarchia precisa, inserendo tag all'interno di altri tag (per chiudere un tag basta porre uno slash davanti al nome del tag che vogliamo chiudere: **</tag>**).

Alcuni esempi di tag utilizzati per la realizzazione dell'applicazione web saranno mostrati di seguito:

- **<html>,** per indicare l'inizio e il tipo di documento.
- **<head>,** per dare informazioni descrittive come il titolo (<title>) oppure sulle fonti utilizzate attraverso link (<link>).
- **<body>,** rappresenta fisicamente il contenuto della pagina, su cui inserire script (<script>) di linguaggio, JavaScript in questo caso, oppure i fondamentali div ( ), che permettono di delimitare determinate porzioni di codice, creando dei veri e proprio contenitori di tag, personalizzando ancora di più l'output.

Altrettanto importanti i servizi che HTML può garantire attraverso questi tag e che sono stati fondamentali per lo sviluppo dell'applicazione web:

- Collegamento verso file di servizio esterni, attraverso il tag <link>
- Inserimento di contenuti interattivi (script).
- Inserimento di pannelli SVG (Scalable Vector Graphics) e bottoni dinamici.
- Diversificazione tra titolo, paragrafo, testo semplice, ecc.

3.5 CSS

Già introdotto nel paragrafo precedente, il CSS (sigla di *Cascading Style Sheets*), in informatica, è un linguaggio usato per definire la formattazione di documenti HTML e XML, ad esempio i siti web e relative pagine web.

L'introduzione del CSS si è resa necessaria per separare i contenuti delle pagine HTML dalla loro formattazione o layout e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione.

La prima pubblicazione da parte del *W3C* è datata 17 dicembre 1996, qualche anno più tardi rispetto al linguaggio "fratello" HTML, ma che da questo momento in poi si è sviluppato parallelamente. L'ultima versione rilasciata è CSS3, accoppiata con HTML5.

I CSS si possono definire come un insieme di modificatori di classi; ogni modificatore viene applicato a specifici elementi della pagina web, identificati attraverso l'uso di *selettori*.

Le classi CSS sono utilizzate per specificare attributi grafici come font, dimensione, colore, spaziatura, bordo e posizione degli elementi all'interno di una pagina web.

I fogli di stile CSS si applicano a cascata: tutte le regole sono valide, ma prevale l'ultima regola, la più specifica.

3.6 JavaScript

Insieme alle due presentate nei paragrafi precedenti JavaScript rappresenta una delle 3 tecnologie sulle quali si fonda lo sviluppo di un portale internet, dove HTML ne gestisce la struttura e CSS lo stile. Ma allora a cosa serve JavaScript?

Originariamente sviluppato da Brendan Eich della Netscape Communications con il nome di Mochan e successivamente di LiveScript, in seguito è stato rinominato "JavaScript" ed è stato formalizzato con una sintassi più vicina a quella del linguaggio Java di Sun Microsystems (che nel 2010 è stata acquistata da Oracle). Fu standardizzato per la prima volta nel 1997, mentre l'ultimo standard risale a giugno 2021.

JavaScript è un linguaggio di programmazione orientato agli oggetti e agli eventi, comunemente utilizzato per la creazione di effetti dinamici interattivi tramite funzioni di script invocate da *eventi* innescati a loro volta dall'utente in diversi modi. Gli effetti che si possono creare sono davvero tantissimi, partendo da immagini animate, fino ad arrivare ad aggiornare contenuti e controllare dati multimediali; ci sono moltissime possibilità e questo lo mette tra i linguaggi di programmazione preferiti dagli utenti.

Lo scopo che assume all'interno di un documento HTML è quello di renderne più dinamico il contenuto e con la crescita costante di internet e della tecnologia generale ha assunto un ruolo fondamentale. L'inserimento delle funzioni di script all'interno di file HTML può avvenire opportunatamente tramite pagine *JSP* o in appositi file con estensione "*.js*".

JavaScript ovviamente non fu l'unica tecnologia del suo genere nei primi anni dalla sua fondazione, infatti dovette affrontare anche la concorrenza delle Applet Java e di Flash. Inizialmente sfavorita, il successo di JS lo si deve con ogni probabilità all'introduzione della tecnica di sviluppo software AJAX, acronimo di Asynchronous JavaScript and XML, che, basandosi su uno scambio dati in background, consente di comunicare in modo asincrono con il server, permettendo così l'aggiornamento dinamico di una pagina web senza esplicito ricaricamento da parte dell'utente. Una caratteristica diventata sempre più importante nel corso degli anni.

Altre motivazioni del suo successo possono essere:

- la semplicità del setup, per cui non c'è bisogno di un ambiente di sviluppo per iniziare e nemmeno di un browser specifico, in quanto JS è supportata da tutti.

- La vastità di librerie (API) e frameworks compatibili, tra cui la nostra *d3.js, JQuery* e molte altre.
- La programmazione end-to-end con *Node.js,* un ambiente di runtime introdotto nel 2009 che permetteva di eseguire JavaScript al di fuori di un browser web, permettendo a JS di girare anche sui server, rendendolo utilizzabile anche per la programmazione server-side. Node.js rappresenta l'idea del paradigma "JavaScript everywhere", unificando tutto lo sviluppo di applicazioni web attorno a un unico linguaggio di programmazione. Ad oggi sono tanti i siti che hanno utilizzato Node.js, tra questi Netflix, Paypal e LinkedIn.
- Una comunità molto attiva, che grazie ad alcuni appositi siti, può sanare dubbi di ogni tipo per un qualsiasi programmatore, da quelli alle prime armi fino ai più esperti.

La sintassi di JavaScript è simile a quella dei linguaggi C e Java, anche se con quest'ultima a causa del nome si è fatta molta conclusione nel corso degli anni. Infatti presenta comunque alcune differenze, come il fatto che il primo sia interpretato e non eseguibile su app, mentre il secondo sia compilato ed eseguibile su app.

Per una corretta immissione del codice scritto in JavaScript all'interno di un file HTML abbiamo 3 modalità:

1. Codice *inline*, che consiste nell'inserire direttamente le istruzioni di JavaScript nel codice di un elemento HTML, assegnandolo ad un attributo che rappresenta un evento. Eccone un esempio:

```
<button type="button" onclick="alert('Ciao!')">Cliccami</button>
```

   Quest'ultima in realtà è considerata una *bad practice* dai programmatori poiché va a "inquinare" parti di HTML con JS.

2. Blocchi di codice, segnalati e inseriti attraverso appositi tag <script> </script>, diventa molto comodo quando devo eseguire codice complesso o lungo.

```
<script>

        // JavaScript goes here

</script>
```

3. JavaScript esterno, che diventa il più consigliato a seconda dell'ambiente in cui si sta programmando. Consiste nel collegare alla pagina HTML, codice JavaScript presente in un file esterno. Questa tencica permette di agganciare script e librerie in modo detto non intrusivo, con il vantaggio di una separazione netta tra la struttura del documento e il codice:

```
<script src="codice.js"></script>
```

Il metodo che ho prediletto, vista l'ampia quantità di codice che ho dovuto scrivere o modificare, e l'ambiente di sviluppo, è stato il numero 2.

CAPITOLO 4 - La libreria D3.js

In questo capitolo sarà data una descrizione della libreria ***d3.js*** di JavaScript che sta alla base di questo progetto e che ha assunto un ruolo decisamente fondamentale per la realizzazione della nostra applicazione web, ovvero la visualizzazione dei dati. Inoltre si tratta della tecnologia meno conosciuta tra quelle presentate fino ad ora e il suo studio ha rappresentato una parte importante di questa tesi e per questo ho deciso di dedicarci un intero capitolo.

Il primo paragrafo è dedicato a un'introduzione generale della libreria, con la presentazione di alcune caratteristiche e aspetti che la contraddistinguono.

Nei restanti paragrafi verranno esaminate le funzionalità che sono state usate per la realizzazione di questo progetto.

4.1 Introduzione a D3.js

D3 è una sorta di acronimo o sigla che sta per Data-Driven Documents, ovvero "documenti creati dai dati". Si tratta dunque di una libreria JavaScript usata per creare visualizzazioni dinamiche ed interattive partendo da dati organizzati (secondo diversi modelli, nel nostro caso tabelle Excel), visibili attraverso un comune browser. Per fare ciò utilizza standard web quali HTML5, SVG E CSS.

Iniziata ad essere sviluppata nel 2011, fu rilasciata per la prima volta nell'agosto di quell'anno (versione 2.0.0), mentre l'ultima versione (la 7.0.0) risale a giugno 2021.

L'utilizzo dei sopra citati standard web ci dà tutte le proprietà di un moderno browser, rendendoci indipendenti da un framework proprietario, combinando insieme una potente visualizzazione dei componenti e un approccio data-driven alla manipolazione DOM (Document Object Model). Con quest'ultimo termine si indica infatti uno standard W3C, che definisce come accedere ai documenti.

Le caratteristiche principali hanno reso questa libreria così popolare e utilizzata sono:

- Orientata ai dati, interagisce facilmente con essi e può supportare diversi formati anche da server remoti.
- Consente di manipolare il DOM, anche dinamicamente.
- Modulare, perciò non devo scaricare tutta la libreria tutte le volte che ho bisogno di una parte di essa, ma solo la parte che mi serve.
- Consente di costruire facilmente e aggiungere al DOM componenti grafiche.
- Agevola animazione e interazione, è infatti disponibile un importante supporto per le animazioni, grazie a funzioni che gestiscono le transizioni tra più stati, permettendo di controllare la durata, il ritardo, è l'attenuazione del movimento. Queste funzioni sono molto reattive agli input dell'utente.

- Il cosiddetto “method chaining”, che permette di connettere funzioni tra loro attraverso dei punti. In questo modo posso avere un codice pulito e compatto.

I vantaggi che invece essa può portare a un programmatore sono:

- Essendo una libreria JS, è facilmente integrabile a interno di documenti HTML attraverso script JS.
- Si focalizza sui dati, quindi è il tool più appropriato e specializzato per la visualizzazione dei dati.
- Libreria open-source, quindi si può lavorare col codice sorgente e aggiungere direttamente le proprie modifiche.
- Lavora con gli standard web.
- Non fornisce alcuna funzione specifica, perciò lascia un controllo completo sulla propria visualizzazione per poterla personalizzare a piacimento.
- Essendo molto leggera, e lavorando direttamente con gli standard web, è estremamente veloce e in grado di lavorare con grandi insiemi di dati.

4.2 Manipolazione degli elementi del DOM

In questo paragrafo illustrerò quali funzionalità della libreria ho utilizzato maggiormente per modificare e manipolare gli elementi del documento HTML. Azioni queste, che nell’ambito di applicazioni server dinamiche volte a permettere la visualizzazione di dati, sono molto comuni e che D3 è in grado di velocizzare e facilitare

La selezione è uno dei concetti chiave per D3 ed è basata su selettori CSS. Ci permette di selezionare uno o più elementi in una pagina web e in più ci permette di modificare, aggiungere in append, o rimuovere elementi in relazioni a un insieme di elementi precedentemente definito.

Per la selezione dunque, D3 utilizza i seguenti due metodi:

1. **select(),** seleziona solo un elemento del DOM, facendo un match con il selettore CSS dato. Se ce ne sono più di uno, riporterà solo il primo.
2. **selectAll(),** seleziona tutti gli elementi del DOM, facendo matching con il selettore CSS dato.

I metodi appena visti selezionano elementi HTML attraverso selettori CSS. In questi selettori si possono definire e accedere ad elementi HTML in 3 modi: con il tag, il nome della classe oppure l’ID di un elemento HTML.

Una volta che l’elemento è stato selezionato lo si può manipolare con diversi metodi, tra cui:

- text(“”): per inserire/modificare il testo dell’elemento selezionato.

- append(“element name”): inserisce un elemento appena prima la fine dell’elemento selezionato.
- insert(“element name”): che inserisce un elemento all’interno dell’elemento selezionato.
- attr(“name”, “value”): che setta il valore dell’attributo specificato.
- style(“name”, “value”): che setta lo stile dell’elemento.

Oltre a queste funzioni, ci sono anche quelle che permettono di animare i suoi elementi.

D3 semplifica il concetto di “animazione” con quello di “transizione” da una forma (o uno stato) ad un’altra. Infatti, il metodo che indica l’inizio di un’animazione è transition().

Di seguito all’invocazione di questo metodo, vengono specificati i parametri che rappresentano lo stato finale che avrà l’elemento al termine della transizione, e possono essere applicate altre funzioni che specificano alcune caratteristiche della transizione. Le più note sono:

- duration(): che specifica la durata della transizione.
- ease(): che specifica la tipologia di movimento durante la transizione.
- delay(): che specifica il ritardo dell’animazione in millisecondi.

4.3 Data Join

Ricordando che D3 è una libreria orientata ai dati e incentrata quindi su di essi, pertanto esisteranno diverse funzionalità che ci aiuteranno nella manipolazione e gestione dei dati. Tutti questi servizi possono essere raggruppati sotto il nome di *Data Join,* anche se possiamo suddividerli più precisamente in due sottogruppi, a seconda se i dati che dobbiamo utilizzare si trovino all’interno di variabili locali oppure in file esterni:

1. **Data Binding**, che si basa sul *binding* (in italiano *legame*) tra i dati che sono stati salvati all’interno di variabili locali (spesso si tratta di Array/Liste) e i diversi elementi del nostro documento HTML, che verranno aggiornati di conseguenza.
   Supporta diversi metodi, tra cui i più importanti sono: data(“nomeArray”) e enter(), strettamente collegati fra loro. Il primo serve per mettere direttamente in relazione il vettore di dati specificato con l’elemento del DOM selezionato, mentre il secondo serve per creare dinamicamente delle references che corrispondono al numero ordinato di un valore dell’array.
2. **Data Loading**, che invece si basa sul *Loading* (in italiano *caricamento*) di dati da file esterni. In questo caso sono disponibili alcuni metodi per manipolare i diversi formati dei file, in particolare ne vediamo quattro:
   - d3.csv()
   - **d3.tsv(),** quello da me utilizzato in questo progetto
   - d3.xml()
   - d3.json()

Tutti questi metodi accettano, una volta chiamati, in input due parametri; il primo per specificare il nome del file da cui voglio andare a leggere i dati (nome che deve essere presente sul file-system o in rete), mentre il secondo per indicare la funzione di *callback,* che voglio eseguire sui dati in questione e su come gestirli di conseguenza.
Più in particolare, d3.csv() accetta file di tipo Comma-Separated Values, ovvero valori separati da virgola, infatti i file CSV sono file di testo che utilizzano le virgole per separare i dati contenuti all'interno di singole celle di una tabella.

D3.tsv() lo vedremo più avanti all'interno di questo elaborato ed è un formato di file simile a quello precedente, ma che separa i valori attraverso tabulazioni.

Gli ultimi due invece accettano rispettivamente formati di tipo XML (Extensible Markup Language) e JSON (JavaScript Object Notation); XML è formattato in modo molto simile a un documento HTML, ma utilizza tag personalizzati per definire oggetti e dati all'interno di ciascun oggetto.
JSON, invece, è un formato adatto all'interscambio di dati fra applicazioni client/server, basato sul linguaggio JavaScript, ma ne è indipendente.

CAPITOLO 5 – Il Progetto

Questo capitolo, che ritengo essere la parte centrale di questo elaborato, nonché la più interessante e personale, è incentrato sulle fasi di vita del progetto iniziato mesi fa in collaborazione con Coopservice, ovvero dall'iniziale studio di d3.js, con la realizzazione di alcuni esempi dinamici di programmazione con tale libreria, fino a uno sviluppo di un'applicazione funzionante, basata sui dati reali raccolti dall'azienda di Reggio Emilia presso l'ospedale Sant'Orsola-Malpighi.

Nei primi paragrafi sarà presentata, oltre al già citato studio della libreria d3.js, la situazione generale che ho trovato quando ho preso in mano questo progetto, già iniziato precedentemente da un collega.

Seguiranno poi alcuni paragrafi riguardanti lo sviluppo in sé della applicazione e le modifiche da me apportate a quella pre-esistente.

Nella parte finale invece tratterò i metodi che ho utilizzato per lavorare al meglio con i dati reali delle tabelle Excel.

5.1 Studio e approccio iniziale a d3.js

Prima di iniziare a fare qualcosa di tangibile che andasse a riguardare direttamente l'applicazione web, ci è stato richiesto di dedicare un po' di tempo allo studio di JavaScript e della libreria d3.js e di provare a realizzare alcuni prototipi, in modo da arrivare più preparati al momento dell'approccio vero e proprio con la programmazione web dinamica. La preparazione infatti è stata di grande aiuto per poter soddisfare le richieste di Coopservice.

La prima cosa che ho fatto, dunque, è stata quella di recuperare alcune lezioni online e diverse documentazioni sull'argomento, in particolare su d3.js, che come spiegato nel capitolo precedente gode di diversi tutorial e anche di una community molto attiva sul web, perciò non è stato troppo difficile reperire ciò che mi serviva.

Il passo successivo consisteva nell'iniziare a provare a scrivere qualche riga di codice utilizzando JS e d3, nel mio caso all'interno dell'editor di codice sorgente Visual Studio Code, provando a realizzare qualche esempio grafico o di figura dinamica, iniziando anche a prendere confidenza con gli *Scalable Vector Graphics* **(SVG),** web standard che ci permette di creare diverse forme come cerchi, rettangoli, ecc., dandoci più flessibilità e potenza in ciò che vogliamo realizzare, e che rappresenta inoltre il formato delle planimetrie relative all'ospedale Sant'Orsola, su cui sarebbero dovuti avvenire gli spostamenti dei pazienti, sotto forma di punti.

SVG è un modo per rendere le immagini in una pagina web. Non è una immagine diretta, ma un mezzo per creare le immagini usando testo e testo. Come suggerisce il nome, Scalable Vector, può modellare in scala sé stesso basandosi sulle dimensioni del browser.

5.2 I primi esempi

Nei primi step di questo progetto, dopo lo studio, è stato richiesto di provare a sviluppare un prototipo di applicazione chiamato “moving dots”, in cui l’obiettivo era quello di plottare dei punti (passati attraverso un array di semplici oggetti, attraverso il data binding) su un SVG di una planimetria di prova molto semplice, controllando il tutto anche con uno slider temporale.

Per implementare questa applicazione sono stati usati diversi metodi della libreria d3; in questo modo ho preso sempre più confidenza con la libreria.

Il primo passo sarà quello di creare un pannello svg, utilizzando la funzione apposita:

```
d3.create("svg");
```

Dopodiché, sempre usando funzioni della libreria d3, posso aggiungere un elemento, in questo caso un cerchio (circle), in append, su un SVG precedentemente selezionato:

```
circle = d3.append("circle")
            .attr("r", 5)
            .attr("cx", points[i].x_in)
            .attr("cy", points[i].y_in);
```

In questo caso il cerchio avrà raggio 5, mentre le coordinate iniziali sono passate attraverso un array locale di punti creato staticamente.

Infine per far muovere i punti verso le coordinate finali, sempre contenute nell’array locale, utilizzerò la funzione transition(), settando duration() a 5000 ms:

```
circle.transition()
            .duration(5000)
            .attr("cx", points[i].x_fin)
            .attr("cy", points[i].y_fin);
```

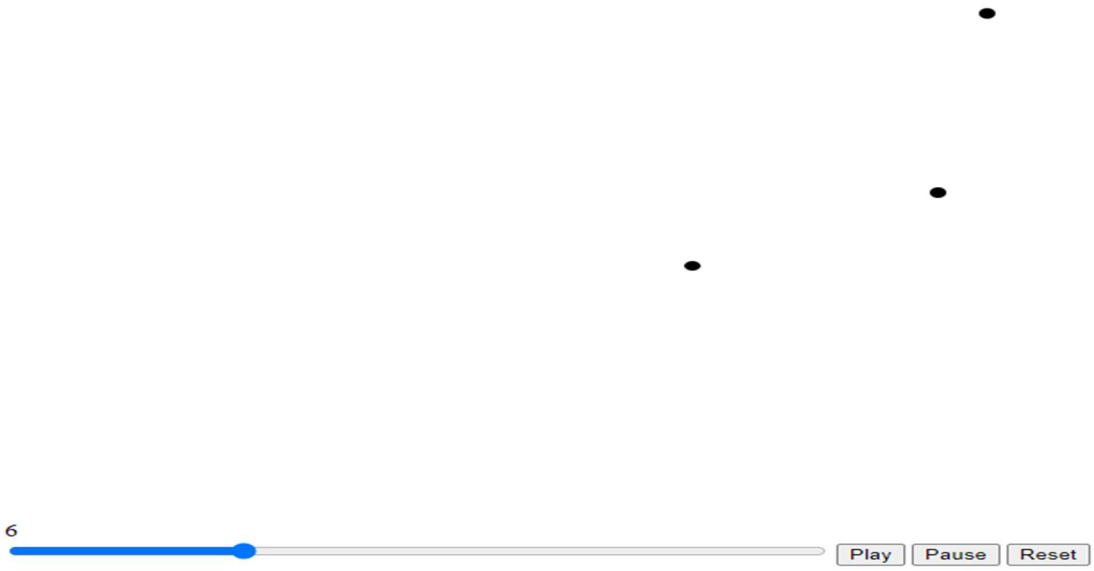
Lo step seguente era quello di aggiungere degli indicatori temporali, due in particolare:

- un campo *time* agli oggetti dell’array, che faccia riferimento a quando deve avvenire la transizione, ovvero lo spostamento.
- Uno *slider* temporale che tenga traccia del tempo trascorso.

Tramite una variabile che viene posta uguale al valore presente sulla barra temporale, si verifica ogni secondo che il campo time di tutti i dots non sia scaduto, e se il punto non è stato mosso (facilmente deducibile utilizzando una variabile booleana), si chiama la funzione move() o moveBack(), a seconda dei casi.

Figura 1: secondo 0, momento in cui i pallini si trovano alla loro posizione iniziale.

I tre bottoni presenti serviranno alle funzioni che sono riportate su di esse. Una volta infatti che viene schiacciato play si azionerà lo slider e la transizione dei pallini, arrivando ad ottenere questo risultato:



5.3 Lo sviluppo dell'applicazione

Dopo aver preso abbastanza confidenza con d3, nei modi spiegati nei paragrafi precedenti, il progetto è stato preso inizialmente in mano dal mio collega studente Gabriele Ruini che insieme al Co-relatore Giovanni Simonini ha concordato i seguenti step:

1. Creare una corrispondenza tra i reparti e le coordinate nell'*SVG*.
2. Inserire attributi più specifici, per consentire di raffigurare i pallini in maniera più esplicativa (ad esempio colorare i pallini in base all'operatore che ha effettuato lo spostamento).

3. Creare una rappresentazione distinta per la visualizzazione dello spostamento dei pazienti e degli operatori.
4. Inserire un date-picker.

Dunque, il mio contributo è iniziato successivamente al completamento di questi 4 passaggi, ed è consistito nella modifica e miglioramento del codice pre-esistente più l'aggiunta di funzionalità che non erano state implementate dal collega. Riassumendo questi sono stati gli step che ho concordato con il Prof Simonini ed eseguito da solo:

1. Implementare una buona navigazione fra i vari padiglioni, piani e reparti dell'ospedale, con l'inserimento di appositi bottoni, garantendo un corretto passaggio dei dati.
2. Andare a intervenire direttamente sui dati reali, aggiungendo un campo di raggruppamento per i reparti che altrimenti sarebbero stati impossibili da mostrare graficamente. Questo sarà poi il campo che verrà mostrato direttamente sulla pagina web.
3. Modificare il codice scritto dal collega, passando da un codice che agiva con dati inseriti staticamente ad uno che utilizza dati reali dinamicamente.
4. Inserimento di metriche e indicazioni relative alle prestazioni dei trasportatori, come richiesto da Coopservice per ottimizzare il lavoro di questi operatori sanitari.

1. Per soddisfare la richiesta di questo primo step, sono partito da una prima rappresentazione molto semplice, che sarebbe poi diventata quella di default una volta avviata l'applicazione, in cui sono presenti i 4 padiglioni dell'ospedale Sant'Orsola, corrispondenti ai dati che ci sono stati consegnati. In particolare quindi, attraverso la funzione, già implementata, che pone le coordinate degli elementi all'interno del SVG, possiamo trovare il Padiglione 1, 2, 23 e 5. Nello stesso modo ho aggiunto abbastanza agevolmente una lista di piani che vanno da -1 a 5, sempre per fare un match corretto col file che contiene tutti i più di 300 reparti. Infine per garantire una buona navigazione fra i diversi padiglioni e piani, ho aggiunto dei bottoni, che, una volta cliccati, chiamano una funzione in grado di rimodellare gli elementi presenti sulla planimetria in base ad apposite variabili che sono modificate dalla funzione attivata in precedenza e che serviranno come filtro per capire cosa mostrare in output. Per tenere traccia di quale situazione ci troviamo, ho aggiunto successivamente delle etichette che potessero garantire all'utente di capire in che punto dell'ospedale si trova in quel momento. Ecco i bottoni che ho aggiunto:
   - *"Visualizza Per Padiglioni",* che rimodella la planimetria mostrando la prima schermata di default, quella con solo i 4 padiglioni, ripulendo eventualmente bottoni in eccesso.
   -*"Visualizza Intero Ospedale",* che rimodella la planimetria mostrando una visuale più ampia, riguardante l'intero ospedale, con la possibilità di vedere i 7 piani della

struttura. Inoltre come il precedente, ripulisce eventuali bottoni in eccesso.
-*"Padiglione N",* con N={1,2,23,5}, questi 4 bottoni vengono mostrati solamente quando si clicca sul primo citato. Essi servono per entrare sul padiglione selezionato e di conseguenza mostreranno i piani di tale padiglione.
-*"Piano N",* con N={-1,0,1,2,3,4,5}, questi 7 bottoni vengono mostrati quando si clicca sul secondo o terzo tipo di bottone. Con questi si entra nella parte più specifica di un ospedale, mostrando reparti relativi a un piano dell'intero ospedale oppure del padiglione selezionato in precedenza.

Per l'aggiunta dei reparti specifici ho dovuto apportare alcune modifiche che spiegherò nel punto seguente.
Riprendendo brevemente la funzione che gestisce le coordinate, essa divide una circonferenza in tante parti uguali quante sono le etichette da rappresentare, ed infine assegna ad essi ascissa ed ordinata con banali calcoli trigonometrici.

2. Per quanto riguarda il secondo punto, già anticipato in parte nei capitoli precedenti, sono andato ad agire direttamente sul file Excel relativo ai reparti, aggiungendo un'informazione, chiamata *Short*, che fornisse una descrizione più breve rispetto a quella data dalla colonna "Reparto / Unità Operativa". Per esempio, lo Short di "PAD1 SOTT - AMB ST10 UROL URODINAMICA BRUNOCILLA PALAGI" è diventato "UROLOGIA", molto più chiaro e leggibile per un utente medio. Oltre a una questione meramente relativa all'utente essa è servita anche per poter avere una resa grafica migliore delle planimetrie, che altrimenti con 300 e più reparti sarebbero state impossibili da realizzare e visualizzare correttamente senza avere sovrapposizioni. Perciò i reparti che avevano la stessa funzione e sono nello stesso piano dello stesso padiglione sono stati raggruppati sotto un'unica *Short*. Dopo questa modifica l'output che è stato mostrato, oltre che corretto e basato su dati reali, ha assunto anche una forma decisamente interessante e sempre più vicina a quello che era stato chiesto.

3. Questo punto è sicuramente stata la parte più difficile del progetto, nonché quella che ha occupato la maggior parte del mio tempo, ma allo stesso modo quella più interessante. Innanzitutto ho utilizzato degli array statici per salvare i padiglioni e i piani, in quanto non avevano un numero ragguardevole, ma anzi, abbastanza contenuto. Per quanto riguarda il salvataggio dei reparti, letti direttamente dalle tabelle Excel, convertite in file TSV (questa parte verrà approfondita nel prossimo paragrafo), ho utilizzato un set creato dinamicamente per poter eliminare gli elementi duplicati, ovvero tutti gli *Short* uguali tra loro. Il riempimento di questo set avveniva dopo controlli appositi (che verificassero in che zona dell'ospedale ci trovavamo) ed era utilizzato direttamente per appendere le etichette sul SVG con il data binding.
Una volta salvate e disposte correttamente le etichette, attraverso la funzione delle

coordinate, si va a leggere dall'altro file Excel, sempre convertito nel formato TSV, che invece conteneva tutti le informazioni relative ai vari spostamenti fra reparti.
Anche in questo caso i dati letti venivano salvati dinamicamente, dopo un controllo relativo alla data selezionata nel *"datarangepicker"*, già inserito precedentemente dal mio collega, all'interno di un array chiamato "moves", contenente 7 campi, relativi alle informazioni sugli spostamenti (date inizio e fine, reparto di partenza e arrivo, trasportatore, ecc).
Dopodiché questo array viene mappato su un altro array, chiamato "nodes", che aggiunge ulteriori campi, tra cui le coordinate iniziali e finali dello spostamento (ricordiamo infatti che i pallini visualizzati sulla planimetria rappresentano direttamente uno spostamento), il raggio e il colore del cerchio, i minuti che dura.
Il passo successivo sarà quello di "forzare" gli spostamenti sul SVG, con la funzione *d3.layout.force()*, aggiungendoli sotto forma di pallini nelle coordinate iniziali già definite. Ovviamente è presente anche un timer, da cui si potrà leggere l'ora e confrontarla con quello del mio spostamento, una volta che queste coincidono allora inizierà la transizione da un ambiente all'altro.
Per semplificare il codice si è scelto di immettere subito tutti gli spostamenti sullo schermo, ed è inoltre bene specificare, che nei casi in cui siamo nella visualizzazione tra padiglioni, molti non saranno visibili poiché avvengono all'interno dello stesso padiglione. Di questi però si tiene comunque traccia nelle metriche dei trasportatori.

4. L'ultimo punto coincide con anche l'ultima parte di codice da me svolta, richiesta esplicitamente dopo una riunione con Coopservice. Una volta realizzato il terzo punto, con la visualizzazione di dati reali, ciò che mancava ancora era un resoconto delle prestazioni dei trasportatori, entità fondamentali in questo progetto. Ciò a cui abbiamo pensato è stato quindi di tenere traccia sia statisticamente che visivamente del lavoro da loro svolto. Perciò la prima cosa che ho realizzato è stata quella di inserire un colore diverso per ogni trasportatore presente nel file. Essendo 42, il colore scelto per ognuno di essi è casuale e cambia ogni volta che l'applicazione viene riavviata.
   Avendo risolto in questo modo la rappresentazione visiva del lavoro dei trasportatori, dove un colore predominante può far capire quale operatore è stato più impegnato, ho pensato di aggiungere a lato, sotto ai bottoni, una lista dinamica che indicasse per le date selezionate il cognome di un operatore, il numero di spostamenti eseguiti e i minuti effettivi di lavoro, ottenuti attraverso un'operazione di sottrazione tra l'ora di fine e inizio trasporto.

Al termine della realizzazione di questi 4 punti ecco ciò che viene mostrato sulla applicazione:

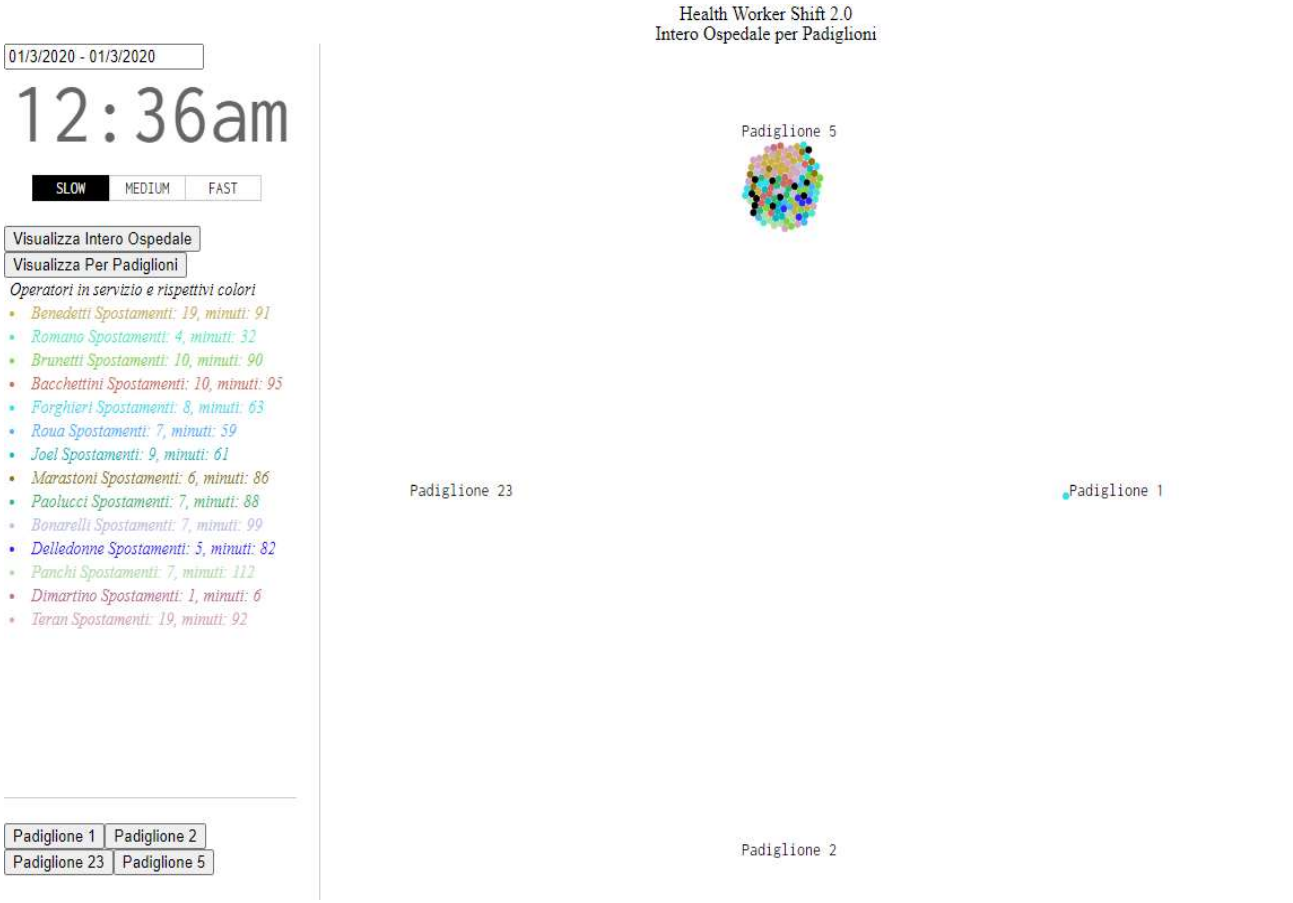


Figura 3. Schermata iniziale in cui vengono mostrati gli spostamenti inter-padiglione. Sulla sinistra è possibile notare un elenco dei trasportatori, con i rispettivi colori e metriche.

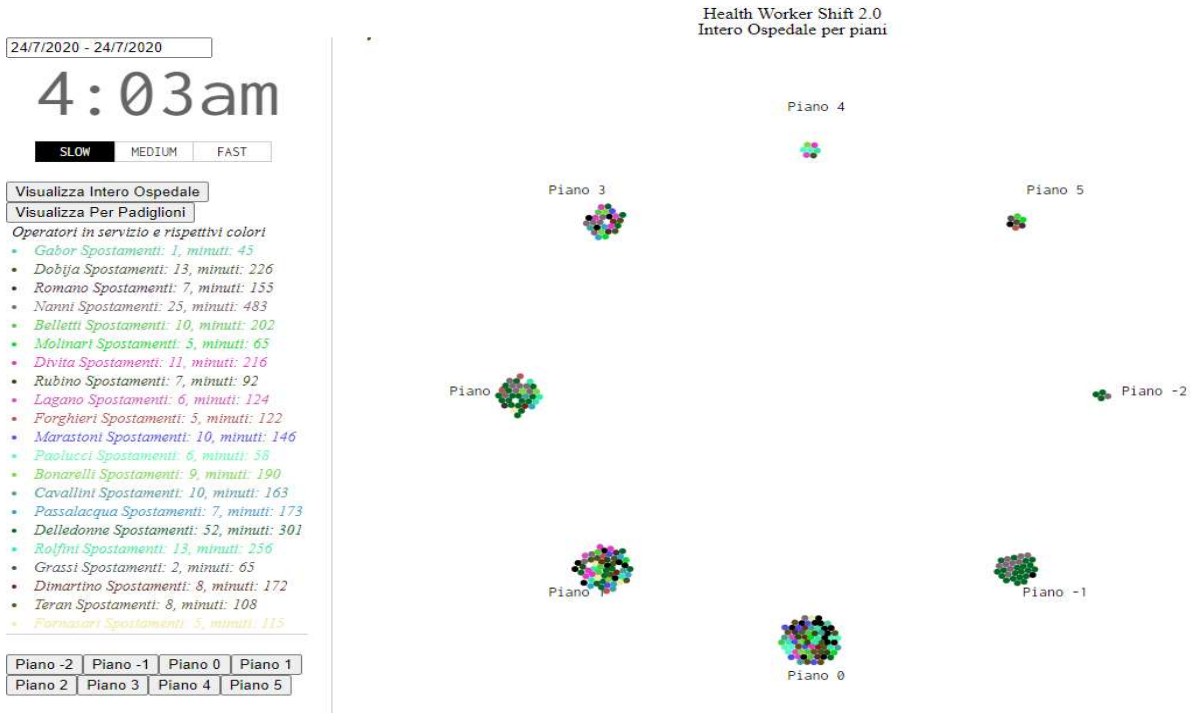


Figura 4. Seconda tipologia di schermata, in cui vengono mostrati gli spostamenti tra piani

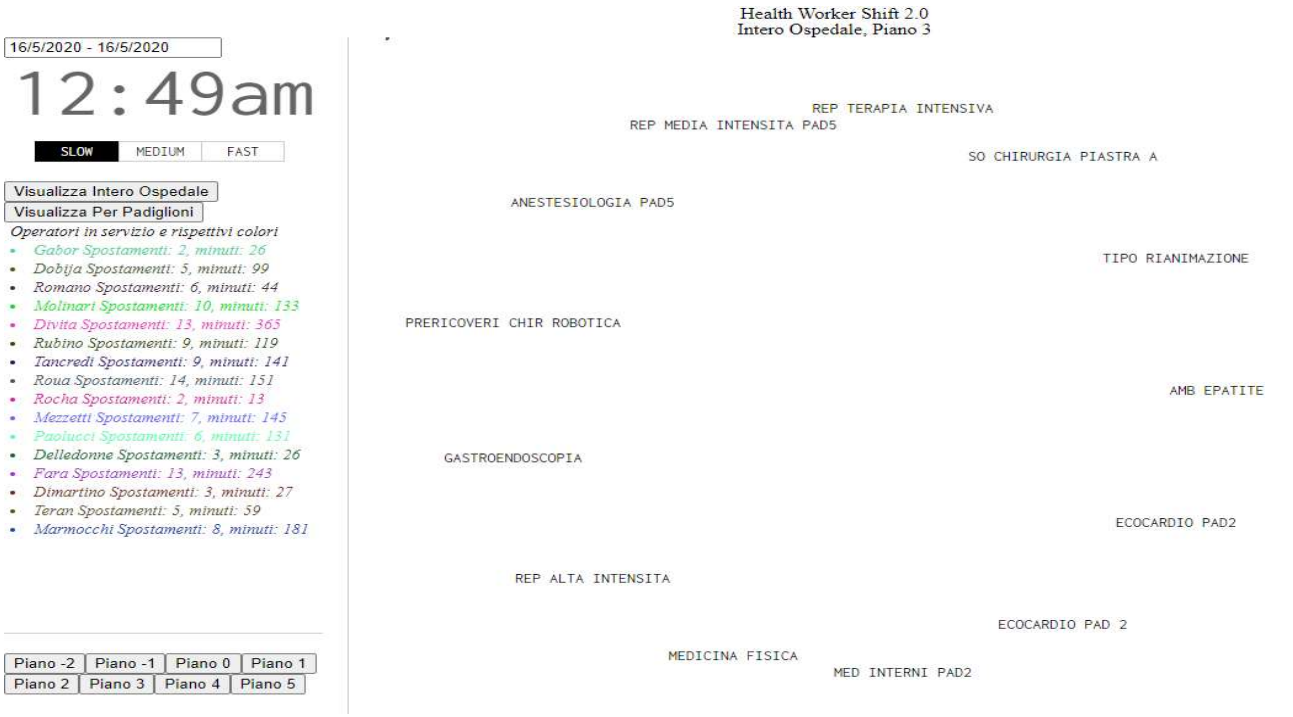


Figura 5. Ultima tipologia di schermata visualizzabile con la navigazione, ovvero quella relativa ai soli reparti.

5.4 Gestione dei dati

Come già anticipato i dati utilizzati ci sono stati consegnati sotto forma di file Excel; tuttavia questo formato risulta incompatibile con la libreria d3 che non prevede un metodo per la sua lettura. Perciò sono stati convertiti nel formato *TSV (Tab-separated value),* dove le righe vengono mantenute mentre le colonne sono separate da una tabulazione.

D3.js prevede un metodo apposito per leggere i dati da file di questo formato, già accennato in precedenza nel paragrafo riguardante il *data loading,* ovvero **d3.tsv(),** che accetta due parametri:

1. *"filename.tsv*", ovvero l'URL o il pathname che identifica il file da cui si vogliono caricare i dati.
2. *function{},* ovvero una funzione di callback, con cui gestire e salvare i dati, con la massima flessibilità.

I dati vengono inseriti, dopo appositi controlli, in un array, attraverso la funzione *push()*, oppure in un set, attraverso la funzione *add().* Si tratta in entrambi i casi di "liste" di oggetti, la cui lunghezza corrisponde al numero di spostamenti; i cui attributi invece corrispondono alle colonne, quindi le informazioni aggiuntive riportate dal file Excel.

Esistono tre funzioni analoghe che permettono di caricare i dati da file di altrettanti formati diversi: CSV, JSON e XML.

Mentre il primo è molto simile a TSV, gli ultimi due formati permettono di eseguire query più complesse.

A differenza del mio collega, il mio compito era quello di cercare di sfruttare al massimo delle loro capacità i dati che ci sono stati forniti da Coopservice e questo infatti è stato l'obiettivo primario di questo elaborato. Oltre alle parti tecniche spiegate nel terzo punto del precedente paragrafo, gran parte del mio lavoro con questi dati, è stata la manipolazione di stringhe (infatti i dati venivano letti di default come tali) e la conversione di queste in diversi tipi di dato.

Ad esempio, visto che per fare i controlli con le date, avevo bisogno di metodi specifici appartenenti alla classe Date, come getTime(), dovevo convertire la stringa relativa a una data in tale tipo. Non bastava però chiamare semplicemente il costruttore, ma anche implementare una funzione che scambiava il mese con il giorno, poiché JavaScript supporta solo la data "all'americana".

Un secondo esempio è invece il l'utilizzo di sottostringhe per verificare in che padiglione o piano mi ritrovo, in quanto non esiste un campo che indichi solamente l'uno o l'altro all'interno del file. Esso infatti è contenuto nel nome completo: "PAD1 PT - ALA A AMB ST2 OCUL VISITE COMPL SCHIAVI PALAGI", dove si può facilmente intuire come sono indicati il padiglione e il piano.

Un terzo ed ultimo esempio riguarda la corrispondenza tra l'array statico contenenti i cognomi degli operatori e la colonna contenente il nome del trasportatore, così salvata: so.cognome. In questo caso, oltre alle solite sottostringhe mi è corso in aiuto anche un metodo che rendesse la prima lettera maiuscola, ottenendo così un match completo con l'array iniziale, primo step verso la colorazione corretta dei pallini e delle metriche.

CAPITOLO 6 – Conclusioni

In questo capitolo illustrerò le potenzialità future dell'applicazione e gli aspetti in cui può essere migliorata, inoltre trarrò le conclusioni sul progetto realizzato e l'esperienza che ho vissuto in questi mesi di programmazione.

6.1 Considerazioni sul progetto

Il lavoro che ho svolto raggiunge quasi la completezza del progetto richiesto da Coopservice. Infatti siamo arrivati a un punto in cui l'applicazione rispetta i dati reali raccolti dall'azienda e li mostra efficacemente tenendo traccia di metriche di performance per i lavoratori coinvolti.

Sicuramente si è trattato di un'esperienza che mi ha aiutato molto a crescere a livello di programmazione web, di cui prima sapevo a malapena le basi, in particolare mi ha fatto conoscere JavaScript e la sua interessante libreria d3.js e approfondire HTML.

Riguardo d3 e l'applicazione si può concludere che:

- Le conoscenze preliminari di HTML e la similitudine di JavaScript con linguaggi di programmazione già studiati e visti, hanno agevolato l'apprendimento di questi.
- Lo studio conseguente della libreria D3, abbastanza semplice e intuitiva, nonché decisamente documentata online, si è rivelato pertanto facilitato.
- L'utilizzo di D3 è stato decisamente idoneo per il compito che era stato richiesto di svolgere, vista la sua orientazione ai dati e alle animazioni.
- Il progetto, che si può dire quasi terminato, può accettare ancora qualche modifica, come un miglior rendimento grafico e l'utilizzo di altre forme per ulteriori rappresentazioni.

6.2 Modifiche future

Visto la realizzazione di un'applicazione in grado di interagire coi dati nella sua interezza, le cose che possono essere aggiunte riguardano perlopiù la grafica e magari ulteriori rappresentazioni oltre ai pallini e alle metriche già inserite.

Un'idea interessante, che avevamo in parte discusso, vista la grande mole di dati, era quella di raggruppare alcuni spostamenti in cerchi più grandi o magari attraverso forme diverse, a seconda che questi vadano nella stessa area. Problema che si presenta se gli spostamenti diventano molti, quindi per finestre temporali abbastanza ampie.

Un'altra idea invece era quella di inserire interfacce per visualizzare meglio il lavoro svolto dagli operatori; queste interfacce si appoggerebbero direttamente alle metriche già presenti e possono assumere la forma di istogrammi, diagrammi, ecc.

Infine un altro aspetto che non è ancora stato esplorato è quello relativo a una diversa colorazione dei pallini, magari in base ad altri informazioni che non sono ancora state sfruttate, come la priorità o il tipo di trasporto.

SITOGRAFIA

Di seguito alcune dei siti Internet consultati nel corso della realizzazione di questo elaborato:

1. Coopservice. www.coopservice.it/
2. Html, JavaScript, Css. www.html.it/ https://stackoverflow.com/
3. D3.js. https://d3js.org/
4. Eclipse. www.eclipse.org/
5. Flowing Data. https://flowingdata.com/
6. Excel. https://it.wikipedia.org/wiki/Microsoft_Excel/
7. Visual Studio Code. https://code.visualstudio.com/

RINGRAZIAMENTI

Si chiude con questo paragrafo l'elaborato e la mia esperienza triennale per ottenere la Laurea in Ingegneria Informatica.

Il raggiungimento di questo traguardo è stata frutto di un percorso intenso, ma allo stesso tempo divertente e interessante, che mi ha sicuramente formato come studente e che mi ha fatto maturare come adulto.

Anche se il titolo è individuale, certamente non avrei potuto conseguirlo senza l'aiuto di alcune persone che mi hanno accompagnato in questi tre anni.

In particolare ringrazio la mia famiglia, mio padre, mia sorella, mio fratello e i miei nonni, che non mi hanno mai fatto mancare l'affetto e il supporto per andare avanti.

Ringrazio Francesca, persona speciale, che mi ha sempre aiutato nei momenti bui e mi ha sempre dato il coraggio che a volte mancava.

Ringrazio sentitamente anche i miei più cari compagni di studi, Gabriele, Paolo ed Emanuele, che hanno reso questa esperienza fantastica e hanno alleviato faticose giornate passate sui libri.

Ringrazio l'ateneo Unimore e i suoi Professori, per avermi dato la possibilità di studiare e formarmi didatticamente.

Infine dedico un ringraziamento a mia madre, il cui pensiero non mi ha mai lasciato e che sarebbe stata molto orgogliosa di questo traguardo.