	<b>SAMT – Sezione Informatica</b>	Pagina 1 di 47
	<b>Gestione Campo Estivo</b>	

## Gestione Campo Estivo

**Titolo del progetto:** Gestione Campo Estivo  
**Alunno/a:** Michea Colautti  
**Classe:** I4AA  
**Anno scolastico:** 2022/2023  
**Docente responsabile:** Guido Montalbetti

1	Introduzione .....	4
1.1	Informazioni sul progetto .....	4
1.2	Abstract .....	4
1.3	Scopo .....	5
2	Analisi .....	6
2.1	Analisi del dominio.....	6
2.2	Analisi e specifica dei requisiti.....	6
2.3	Use case.....	11
2.4	Pianificazione .....	12
2.5	Analisi dei mezzi.....	13
2.5.1	Software .....	13
2.5.2	Hardware.....	13
3	Progettazione.....	14
3.1	Design dell'architettura del sistema.....	14
3.2	Design dei dati e database .....	14
3.3	Design delle interfacce .....	17
3.3.1	Interfacce di livello 1 .....	17
3.3.2	Interfacce di livello 2.....	19
3.3.3	Interfacce di livello3.....	20
3.4	Design procedurale .....	22
3.4.1	Diagramma di flusso ospite .....	22
3.4.2	Diagramma di flusso volontari ed infermieri .....	23
3.4.3	Diagramma di flusso amministratori .....	24
4	Implementazione .....	25
4.1	Il Database .....	25
4.1.1	Tabella Person .....	25
4.1.2	Entità legate a ospiti .....	25
4.1.3	Entità legate a volontari.....	26
4.1.4	Entità legate ad amministratore.....	26
4.1.5	Entità generiche .....	27
4.2	Sito web .....	27
4.2.1	Creazione delle views .....	27
4.3	Laravel.....	28
4.3.1	Setup del progetto Laravel .....	28
4.3.2	Trasferimento delle views.....	29
4.3.3	Login in Laravel .....	30
4.4	Realizzazione del progetto .....	32
4.4.1	Home principale .....	32
4.4.2	Home campo .....	32
4.4.3	Registrazione e login.....	32
4.4.4	Registrazione al campo.....	32
4.4.5	User page.....	34
5	Test.....	36
5.1	Protocollo di test .....	36
5.2	Risultati test.....	44
5.3	Mancanze/limitazioni conosciute .....	44
6	Consuntivo.....	45
7	Conclusioni .....	45
7.1	Sviluppi futuri .....	46
8	Bibliografia .....	47
8.1	Sitografia .....	47
9	Allegati .....	47

Figura 1 Use Case .....	11
Figura 2 Gantt preventivo .....	12
Figura 3 ER - Versione 1 .....	14
Figura 4 ER - Versione 2 .....	16
Figura 5 <i>Home page</i> sito.....	17
Figura 6 <i>Home page</i> campo .....	18
Figura 7 Pagina utente personale.....	18
Figura 8 Pagina esplorativa .....	19
Figura 9 Pagina utente per infermiere .....	19
Figura 10 Pagina utente personale admin.....	20
Figura 11 Pagina modifica campo .....	20
Figura 12 Pagina disponibilità.....	21
Figura 13 Pagina utente amministratore.....	21
Figura 14 Diagramma di flusso ospiti.....	22
Figura 15 Diagramma di flusso volontari e infermieri.....	23
Figura 16 Diagramma di flusso amministratori .....	24
Figura 17 Struttura sito .....	29

## 1 Introduzione

### 1.1 Informazioni sul progetto

In questo capitolo raccogliere le informazioni relative al progetto, ad esempio:

- Allievi coinvolti nel progetto: Michea Colautti
- Classe: I4AA Scuola Arti e Mestieri Trevano
- Docenti responsabili: Guido Montalbetti
- Data inizio: 29 agosto 2022.
- Data di fine: 07 dicembre 2022.
- Linguaggio: PHP
- Framework: Laravel 9

### 1.2 Abstract

*Questo progetto nasce dalla necessità di creare un sito per la gestione di un campo estivo che sia facile da utilizzare e efficiente per lo scopo per il quale sarà creato. Non si tratta di un campo estivo qualunque rivolto a bambini e/o adolescenti, ma pensato per le persone anziane, o con disabilità, per assaporare l'esperienza di un campo estivo in tutta sicurezza, con personale qualificato, e durante il quale vengono messe in atto tutte le speciali attenzioni richieste.*

*Il sito prevederà la possibilità di inserire, con i dati personali degli utenti, certificati medici e dichiarazioni di salute puntuali, fornendo così al personale del campo un quadro completo degli utenti presenti. In questo modo il personale potrà concentrarsi al 100% sugli ospiti. Non sarà solo l'aspetto relativo alle iscrizioni ad essere sviluppato -con conseguente miglioramento dell'accoglienza- dato che il sito dedicherà agli ospiti (o ai famigliari che si occuperanno di iscriverli) un'attenzione particolare rispetto alle esigenze/preferenze dietetiche; si potranno indicare allergie ed intolleranze, in modo che il personale della cucina potrà così offrire un menu sicuro e variegato.*

*Durante tutto il campo gli ospiti beneficeranno inoltre di un servizio lavanderia, ed anche i trasporti da e per il campo saranno presi a carico se gli ospiti ne indicheranno la necessità.*

*Nel sito sarà inoltre presente un'interfaccia accessibile unicamente al personale medico del campo che potrà consultare le informazioni riguardanti lo stato di salute (malattie, prescrizioni mediche, allergie, e qualsiasi dato immesso al momento dell'iscrizione).*

*This project is born out of the necessity to create an easy and efficient website to handle the management of a summer camp, dedicated to elderly or disabled people. It's a great opportunity for them to enjoy the experience of a summer camp in complete safety, with qualified medical staff and special care.*

*The site will provide the possibility of entering, with the personal data of the users, medical certificates as well as accurate health declarations, which will ease the whole work of the camp staff, who will thus be able to concentrate 100 per cent on the guests. It is not only the process of enrollment that will be developed - making the guests properly accommodated- as throughout the enrollment process the guests (or their relatives) will also be in the position of reporting special needs concerning diets and intolerances: the kitchen staff will thus be able to offer a safe and varied menu.*

*Throughout the camp, guests will also benefit from a laundry service, and transport to and from the camp will also be managed if the guests or their relatives will book them.*

*The site will feature an interface only accessible to the camp's medical staff, where they will be allowed to consult all health data provided (illnesses, medical prescriptions, allergies, and any data provided during the enrollment process).*

### 1.3 Scopo

Lo scopo del progetto è quello di creare un gestionale per un campo estivo per persone disabili e/o anziane. Il gestionale sarà composto da un sito, messo online entro 2 mesi dall'inizio del progetto, che sarà utile sia agli amministratori sia agli utenti del campo stesso.

Una volta raggiunto il sito, una pagina *home* con le informazioni del campo e un collegamento al “Centro Diurno della Fondazione Vita Serena”, la fondazione che organizza il campo accoglierà l'utente. Sarà possibile registrarsi o effettuare un *login*.

Una volta loggati o registrati si potrà procedere con l'iscrizione al campo estivo:

Se si è già iscritti, appariranno le informazioni del campo così come una lista degli utenti già iscritti. Essa sarà filtrabile e ordinabile tramite una serie di attributi, come ad esempio il nome, cognome, la mansione, ecc.

In base a che ruolo ha l'utente, saranno mostrate più o meno informazioni sugli utenti.

La lista di base degli utenti sarà però uguale per tutti.

La differenza starà nelle informazioni alle quali si potrà accedere cliccando un utente nella lista.

- Un ospite potrà vedere solo i dati anagrafici di tutti gli altri utenti
- Un volontario potrà vedere i dati anagrafici di tutti gli utenti e le intolleranze alimentari
  - Ciò è pensato per i volontari in cucina che sapranno quali alimenti evitare.
- Un volontario al quale sono riconosciute conoscenze infermieristiche, potrà vedere tutti i dati di tutti gli utenti, inoltre potrà modificare i dati medici.
- Un amministratore potrà vedere e modificare tutti i dati di tutti gli utenti, eccetto quelli medici, che sono riservati al personale medico o ai volontari abilitati in tal senso.
  - Un amministratore gestisce i dati degli utenti del campo e se necessario ha la facoltà di eliminare un utente dalla lista,

Se invece non si è ancora iscritti al campo estivo ci sarà un collegamento al form di iscrizione.

Qui sarà possibile selezionare per quale posizione si desidera iscriversi: volontario o ospite.

Nel caso degli ospiti essi (o chi per essi) dovranno completare tutti i campi richiesti, compresi i formulari che certificano lo stato di salute ed eventuali medicinali e posologia.

Dal *form* sarà possibile scaricare l'elenco del corredo e le regole del campo, che andranno accettate per procedere con l'iscrizione.

Nel caso di un volontario, per le persone minorenni, oltre al *form* bisognerà compilare un certificato medico sullo stato di salute. Per i maggiorenni non è obbligatorio ma consigliato. In seguito il volontario potrà scegliere la mansione che preferisce, o potrà anche lasciare questo campo in bianco; dopo di che, in seguito ad una discussione con l'amministratore, gli sarà assegnata una mansione.

Gli account degli amministratori saranno creati da altri amministratori.

Un amministratore potrà inoltre modificare le informazioni dei campi precedenti e di quello attuale. Potrà anche copiare la documentazione degli scorsi campi nella pagina di quello nuovo, o apporre eventuali modifiche.

## 2 Analisi

### 2.1 Analisi del dominio

Ad oggi esistono molti campi estivi. Tuttavia, poter beneficiare un'esperienza "personalizzata" come quella che viene offerta nel campo estivo di questo progetto appare importante. Trovo infatti che persone anziane o disabili spesso non possano fare tutto quello che una persona in salute può fare, spesso per motivi puramente pratici: a volte capita infatti che per semplici difficoltà organizzative si chiudano molte porte a persone più fragili per ragioni anagrafiche o per disabilità. Tramite il mio progetto esse avranno la possibilità di partecipare ad un campo estivo attento ai loro problemi e alle loro necessità. I formulari di iscrizione e la presenza online di una pagina dedicata riguardante lo stato di salute (che sarà visibile al personale medico) permetteranno un'esperienza sicura e piacevole, che idealmente li metterà al riparo da cattive sorprese. Oltre alla parte riguardante l'aspetto medico, il mio progetto permetterà anche di visualizzare una pagina in cui saranno presenti tutti gli ospiti e, per ognuno di essi, eventuali intolleranze o esigenze alimentari. Gli utenti saranno quindi infermieri, ospiti, e volontari: questi ultimi aiuteranno nella gestione interna del campo, ma anch'essi potranno segnalare eventuali allergie nel processo di iscrizione, se fosse necessario. Nel sito da me creato ovviamente anche gli amministratori potranno interfacciarsi. Il sito permetterà agli amministratori di evitare di avere a che con certificati cartacei e formulari riempiti in modo errato, permettendo una gestione molto più comoda e veloce. La presenza di funzioni come ad esempio potere ordinare gli ospiti in base a più criteri, sarà un passo avanti rispetto al passato.

### 2.2 Analisi e specifica dei requisiti

ID: REQ-001	
<b>Nome</b>	Creazione pagina base
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Creazione della pagina home con collegamento a fondazione.
Sotto requisiti	
<b>001</b>	Mostrare anteprima informazioni campo estivo corrente

ID: REQ-002	
<b>Nome</b>	Creazione maschera registrazione
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Collegamento in schermata home

ID: REQ-003	
Nome	Creazione maschera <i>login</i>
Priorità	1
Versione	1.0
Note	Collegamento in schermata home

ID: REQ-004	
Nome	Creazione Form iscrizione al campo.
Priorità	1
Versione	1.0
Note	Un ospite o un volontario, specificando il loro ruolo, si iscrivono al corso fornendo tutti i dati.
Sotto requisiti	
001	L'account rimane in sospeso fino all'approvazione da parte di un amministratore.

ID: REQ-005	
Nome	Creazione pagina personale ospite loggato iscritto al corso
Priorità	1
Versione	1.0
Note	Una pagina dove l'utente ha la possibilità di vedere e modificare i suoi dati inerenti al campo.

ID: REQ-006	
Nome	Creazione pagine esplorative per ospite iscritto al campo.
Priorità	1
Versione	1.0
Note	Un ospite può vedere, oltre ai dati sul campo, i dati anagrafici delle altre persone al campo. Con la loro mansione.
Sotto requisiti	
001	Lista con utenti, filtrabili per nome, cognome, funzione.
002	Ogni utente ha una sua pagina, cliccando sull'utente viene aperta. Contiene tutte le informazioni che l'ospite ha il permesso di vedere.

ID: REQ-007	
<b>Nome</b>	Creazione pagine per volontario loggato iscritto al campo
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Una pagina dove il volontario ha la possibilità di vedere e modificare i suoi dati inerenti al campo: come la sua disponibilità.

ID: REQ-008	
<b>Nome</b>	Creazione pagine esplorative volontario loggato iscritto al campo
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Un volontario può vedere, oltre ai dati sul campo, i dati anagrafici delle altre persone al campo. Con la loro mansione. Inoltre il volontario vede le allergie di tutte le altre persone al campo.
Sotto requisiti	
<b>001</b>	Lista con utenti, filtrabili per nome, cognome, funzione.
<b>002</b>	Ogni utente ha una sua pagina, cliccando sull'utente viene aperta. Contiene tutte le informazioni che l'ospite ha il permesso di vedere.
<b>003</b>	Se ha mansione "Infermiere" può visualizzare e modificare informazioni mediche.

ID: REQ-009	
<b>Nome</b>	Creazione pagina per amministratore loggato.
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Una pagina dove l'amministratore ha la possibilità di vedere e modificare i suoi dati.

ID: REQ-010	
<b>Nome</b>	Creazione pagine di modifica utenti dei dati per amministratore.
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Un amministratore, selezionando un utente dalla lista, e visualizzandolo, può modificare i suoi dati.
Sotto requisiti	



<b>001</b>	Lista con utenti, filtrabili per nome, cognome, funzione.
<b>002</b>	Ogni utente ha una sua pagina, cliccando sull'utente viene aperta. Contiene tutte le informazioni che l'ospite ha il permesso di vedere.

ID: REQ-011	
<b>Nome</b>	Creazione pagine di modifica dati campo per amministratore.
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Un amministratore, può modificare i dati sul campo corrente o su quelli passati. Inoltre può creare una nuova edizione del campo, inserendo tutte le informazioni necessarie.

ID: REQ-012	
<b>Nome</b>	Creazione pagina per la creazione di un utente da parte di un amministratore.
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Un amministratore può autonomamente creare un account per un ospite o un infermiere.

ID: REQ-013	
<b>Nome</b>	Creazione funzione di approvazione di un volontario.
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Un amministratore può attivare o eliminare l'account di un volontario o di un ospite. Riferito a requisito 005.001.

ID: REQ-014	
<b>Nome</b>	Pagina per visualizzare disponibilità volontari
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Un amministratore può visualizzare la disponibilità dei suoi volontari

ID: REQ-015	
Nome	Pagina per modifica informazioni campo
Priorità	1
Versione	1.0
Note	Un amministratore può cambiare le informazioni del campo

ID: REQ-016	
Nome	Sito responsive
Priorità	2
Versione	1.0
Note	-

## 2.3 Use case

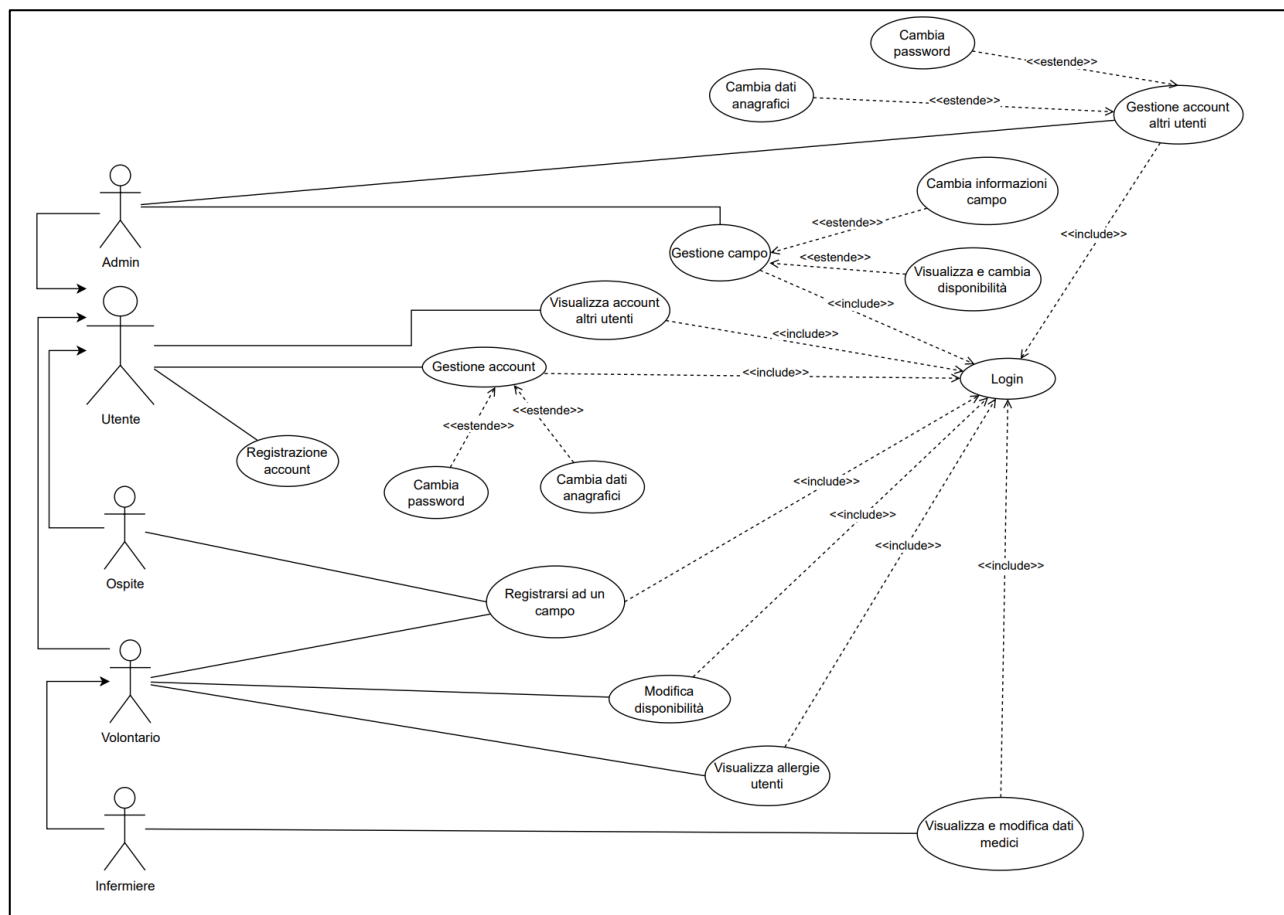


Figura 1 Use Case

Questo *Use Case* rappresenta tutte le funzioni della mia applicazione: l'utente base rappresentato "Utente", è soltanto una base per meglio rappresentare il tutto, non è dunque da considerare un utente finale. La prerogativa per buona parte delle funzioni è, logicamente, essere autenticati.

Essenzialmente tutti gli utenti del campo estivo possono fare tutto ciò che è collegato all'Utente: Con "gestione account" si intende una gestione completa, che permette di cambiare i dati anagrafici e la password. La registrazione account avviene una volta sola e differisce dalla registrazione al campo.

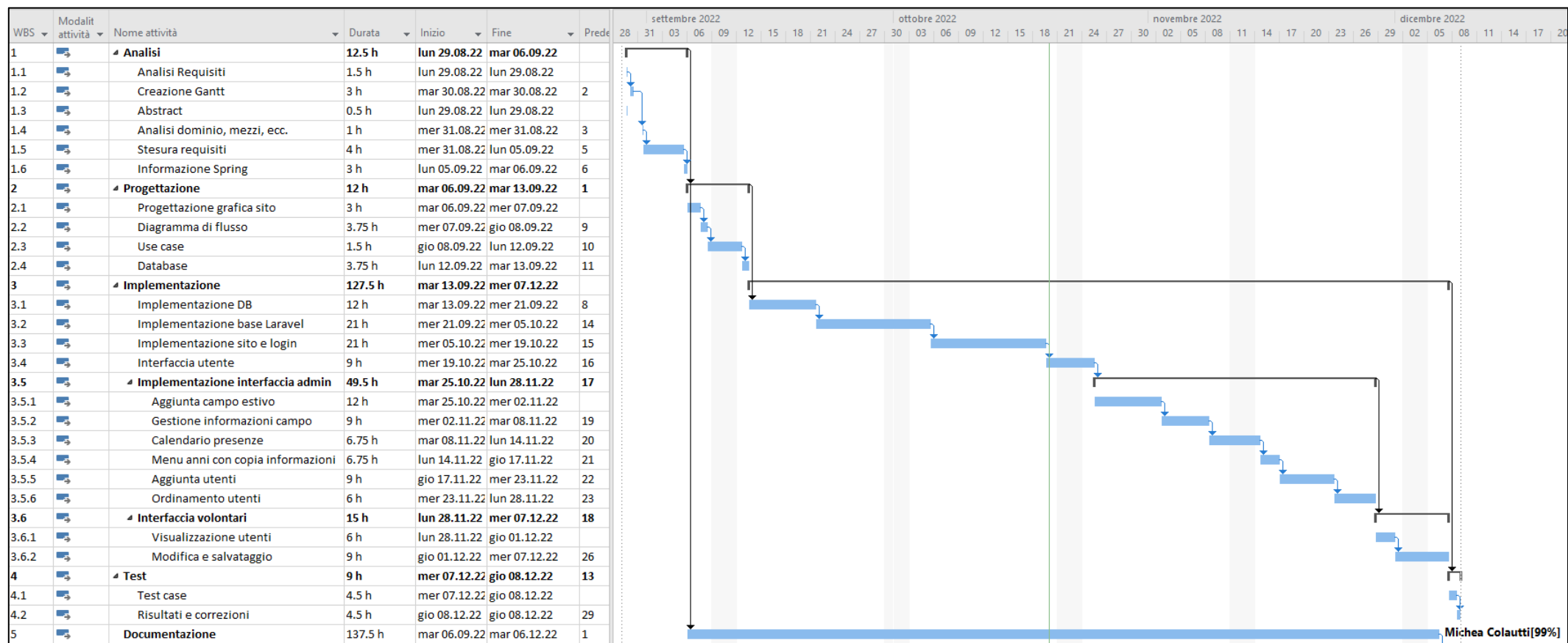
Sia l'ospite che il volontario -una volta che si sono autenticati- possono registrarsi al campo estivo.

Un volontario può modificare la sua disponibilità e visualizzare le allergie degli utenti.

Un infermiere, in aggiunta a tutte le funzioni del volontario, può gestire i dati medici di tutti gli utenti nel campo.

Infine un amministratore può gestire tutti gli account del sito e cambiare le informazioni del campo.

## 2.4 Pianificazione



**Titolo del progetto:** Gestione Campo Estivo  
**Alunno/a:** Michea Colautti  
**Classe:** I4AA  
**Anno scolastico:** 2022/2023  
**Docente responsabile:** Guido Montalbetti

## 2.5 Analisi dei mezzi

### 2.5.1 Software

- MockFlow 1.4.7
- Draw.io
- Gitlab
- GitHub Desktop 2.9.12
- Laravel 9
- VS Code 1.7
- Nicepage desktop
- nicepage.com
- 

### 2.5.2 Hardware

- PC scolastico
- MacBook Air personale da remoto

**Titolo del progetto:** Gestione Campo Estivo  
**Alunno/a:** Michea Colautti  
**Classe:** I4AA  
**Anno scolastico:** 2022/2023  
**Docente responsabile:** Guido Montalbetti

### 3 Progettazione

Essendo consapevole che la progettazione è una fase importante di ogni progetto, vista l'esperienza accumulata in questi anni, ho voluto dedicare il tempo necessario ad essa, definendo tutti gli aspetti sui quali ho riflettuto.

#### 3.1 Design dell'architettura del sistema

L'architettura del mio sito è relativamente semplice. Alla base c'è Laravel: un framework web per lo sviluppo in PHP, basato sul modello di sviluppo MVC. Laravel è stato creato per facilitare l'avvio e la gestione di applicazioni in PHP.

#### 3.2 Design dei dati e database

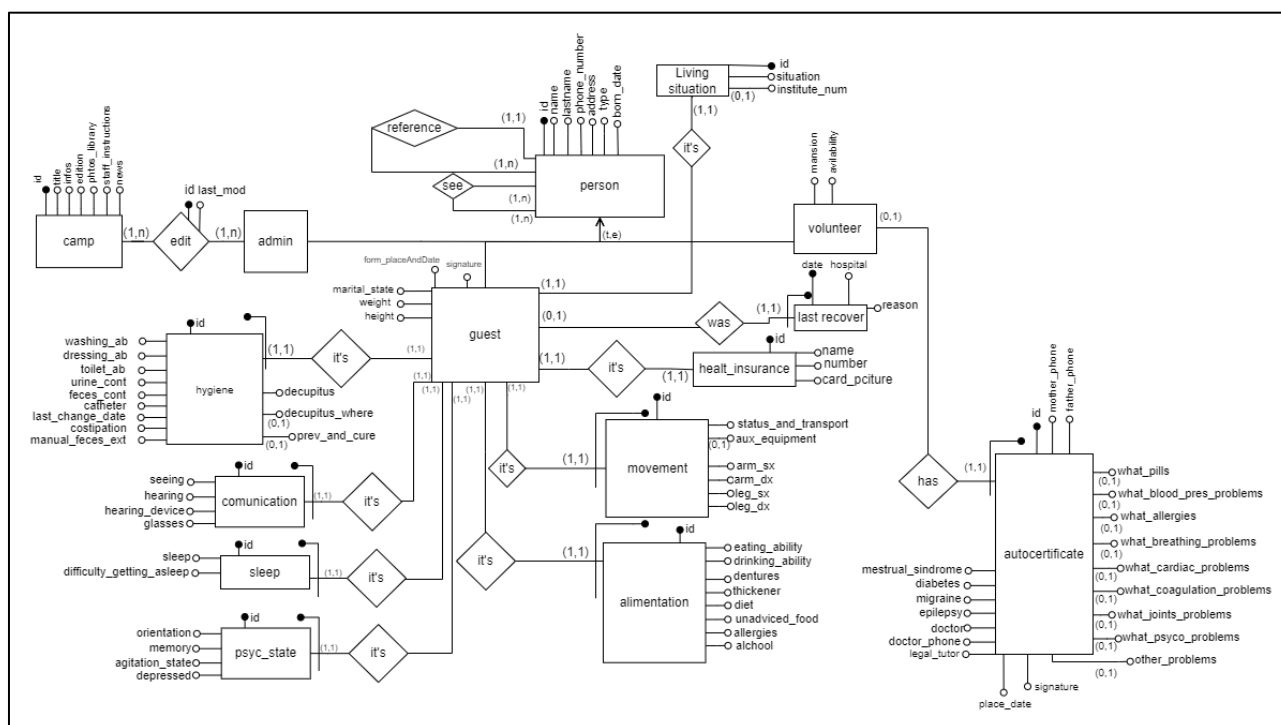


Figura 3 ER - Versione 1

Per realizzare lo schema ER ho studiato la procedura di iscrizione attuale tramite i documenti che mi sono stati forniti, e ho riflettuto su come avrei potuto implementare il salvataggio dei dati. Come si vede dallo schema, ho proceduto in maniera modulare. Inizialmente ho creato uno schema ER con le convenzioni standard della scuola. Ho riflettuto, in particolare, sul concetto di evitare di utilizzare tabelle eccessivamente grandi e con molti campi *null*. Ho quindi optato per generalizzare gli utenti in un'unica entità, "person". La specializzazione "totale, esclusiva" indica che tutti gli utenti si devono classificare come una delle tre sotto entità elencate e che non possono essere due cose contemporaneamente.

#### Entità person

Questa è l'entità base, possiede alcuni campi che tutti gli utenti hanno in comune, come il nome, cognome, e id, che è la *primary key* per gli utenti nell'intero DB. Un altro campo importante è **type**, ovvero tipo. Questo dato serve a distinguere la tipologia di utente: **admin**, **guest**, **volunteer**.

Le uniche relazioni che coinvolgono l'entità **person** è **reference**, che indica la persona di riferimento, e **see**, che rappresenta la possibilità di tutti gli utenti di vedere qualunque altro utente.

### Entità admin

L'amministratore è l'entità più semplice del DB: esso può -oltre alle funzioni basi di **person**- modificare i dati di un campo. Le modifiche vengono storicizzate nella relazione **edit**.

### Entità guest

L'ospite può sembrare un'entità complessa, ma in realtà, una volta capito il meccanismo, è abbastanza semplice.

Seguendo il modello del formulario di iscrizione, che è diviso a capitoletti in base a quello che viene chiesto, ho creato diverse tabelle che memorizzano tutti i dati degli utenti. Come si vede dall'identificatore esterno, tutte queste tabelle avranno come *foreign key* l'id del **guest**, che viene logicamente ereditato da **person**.

Ogni tabella memorizza quindi un set di dati che hanno un argomento comune. Le tabelle che usano questo meccanismo sono:

- *hygiene* → L'igiene personale dell'utente e le sue necessità nella pulizia personale e nelle funzioni corporali.
- *communication* → Le sue abilità di comunicazione
- *psyc\_state* → Lo stato psichico
- *alimentation* → Come l'ospite si alimenta e se ha necessità particolari
- *movement* → Le abilità motorie dell'ospite
- *last\_recover* → Il suo ultimo eventuale ricovero in ospedale
- *healt\_insurance* → Le informazioni sull'assicurazione sanitaria.

### Entità volunteer

Il volontario ha soltanto 2 campi in più rispetto all'entità **person**, tuttavia in maniera simile al **guest** ha anch'egli un'entità separata per memorizzare i dati medici. Questa è l'autocertificazione che il volontario deve obbligatoriamente compilare, se minorenne. Ci sono molti campi 0-1, questo perché nel formulario bisognava prima spuntare un *checkbox* per dire se era presente un disturbo e in seguito specificarlo. Ho pensato di semplificare i campi immettendo un campo facoltativo in cui sia possibile inserire il disturbo. Questo schema è molto esplicativo per quanto riguarda la struttura e la filosofia che ho impiegato per realizzarlo, tuttavia non è esplicativo per quanto riguarda l'implementazione. Perciò, di seguito, è presente una seconda immagine, creata con convenzioni diverse ma che completa la spiegazione.

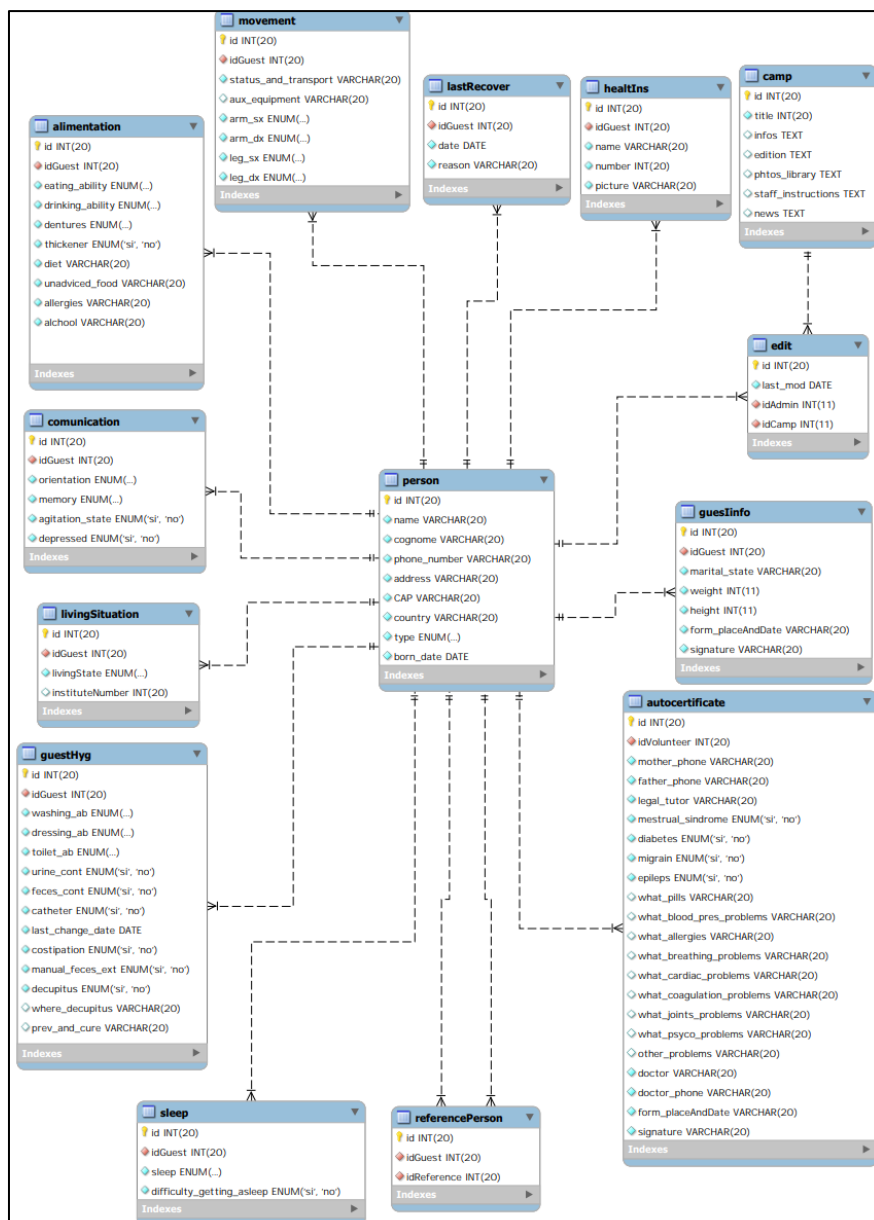


Figura 4 ER - Versione 2

Questo schema è molto più esplicativo per quanto riguarda la struttura “tecnica” del database.

Le entità sono diminuite, dato che la specializzazione non viene rappresentata. Tutti i campi presenti sull’entità **person** sono rimasti, e tutti gli altri sono stati divisi in tabelle.

Nelle tabelle di appoggio sono presenti degli id, come **idGuest** o **idVolunteer**. Questi non sono altro che *foreign key* che fanno riferimento all’*id* nella tabella **person**.

Per il resto il DB rimane identico a prima.

Da notare che molti campi sono degli *enum*.

Ho deciso di adottare questo tipo di dato poiché nel *form* originale c'erano punti in cui bisognava scegliere tra una serie di opzioni: per esempio, nella sezione movimento bisognava selezionare per ogni arto il tipo di mobilità e le possibilità erano:

- Buona
- Ridotta
- Nessuna

Queste opzioni sono riportate nel campo *enum* da me creato.



### 3.3 Design delle interfacce

Per questo capitolo non ho realizzato interfacce molto complesse o articolate. Questo perché ho voluto concentrarmi sulle funzionalità. Voglio sfruttare il fatto che il sito deve essere *responsive* per utilizzare un *template* adeguato e adattarlo alle necessità del sito.

Ho comunque realizzato quella che reputo essere una buona ossatura per sapere in che direzione sviluppare il sito; ho scelto di mantenere lo stesso stile e struttura per più pagine, poiché lo scopo deputato è a grandi linee lo stesso. Le pagine utenti di amministratori o di utenti normali saranno dunque pressoché identiche, fermo restando che gli admin avranno qualche controllo in più. Anche le pagine che mostreranno i dati degli altri utenti del campo saranno uguali per tutti. L'unica differenza sarà la quantità di dati che si vedranno e la possibilità di modificarli o meno.

#### 3.3.1 Interfacce di livello 1

In questo capitolo sono raccolte le interfacce pubbliche o, in generale, accessibili e uguali per gli utenti nel campo. Ci sono casi in cui le interfacce cambiano leggermente in base all'utente autenticato.



Figura 5 Home page sito

Questa è la *home page* del sito internet. Come accennato nello scopo il progetto è per la fondazione “Vita Serena”, che gestisce anche un centro diurno. Per questo nella *home page* del sito ci devono essere due collegamenti, che portano al sito del Centro Diurno e al Campo estivo di Olivone.

Per il contenuto mi sono basato su quello che è attualmente il sito della fondazione, ma lo stile cambierà.

Questa è la *home page* del Campo Estivo.

Oltre ai collegamenti per le informazioni della fondazione, presenti anche nella home page generale, qui è possibile registrarsi o autenticarsi al sito tramite un *pop-up*.

Una volta registrati sarà possibile iscriversi al campo tramite un *form*.



Figura 6 Home page campo

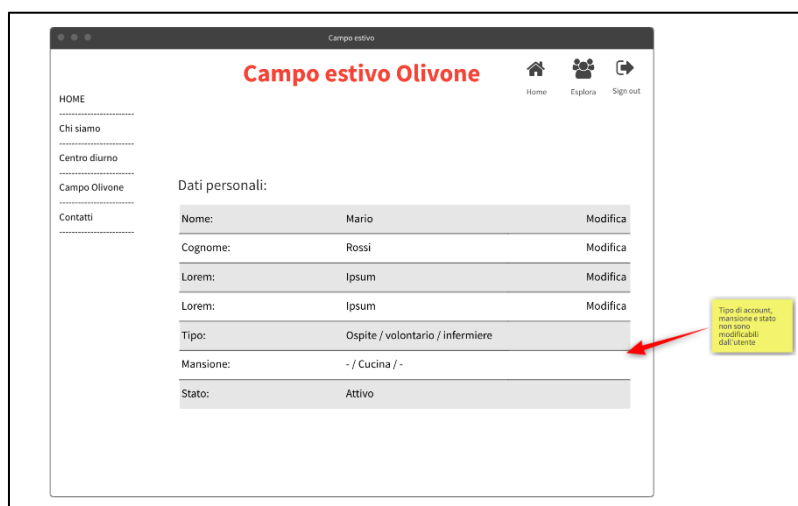


Figura 7 Pagina utente personale

Quando si è riempito il *form* di iscrizione si viene reindirizzati alla pagina persona. Qui l'utente può modificare i suoi dati personali. Gli unici campi non accessibili sono il tipo di account, la mansione e lo stato.

La mansione è un campo che viene riempito solo se l'utente è un volontario. Da questa pagina si può accedere alla sezione esplorativa del sito, ma solo se l'account è stato attivato.

Questa è la pagina esplorativa degli utenti del campo. È uguale per tutti.

In alto, con le icone, ci si può muovere nel sito. Solo l'amministratore avrà però accesso alla pagina dove sono elencate le disponibilità degli utenti.

Usando la barra di ricerca e il *pop-up* contenente i filtri si possono cercare uno o più utenti.

Cliccando sull'intestazione di una colonna i dati verranno ordinati in ordine crescente o per tipo. Cliccando nuovamente verranno ordinati al contrario.

Selezionando un utente si apre la sua pagina personale, che sarà praticamente identica a quella dell'account personale.

In base all'utente che la visualizza verranno mostrate più o meno informazioni.

- Un ospite vede solo i dati anagrafici.
- Un volontario, in aggiunta, vede le allergie.
- Un infermiere, in aggiunta, vede e può modificare i dati sanitari.
- Un amministratore vede e può modificare tutti i dati, **eccetto quelli sanitari, che sono nascosti**.

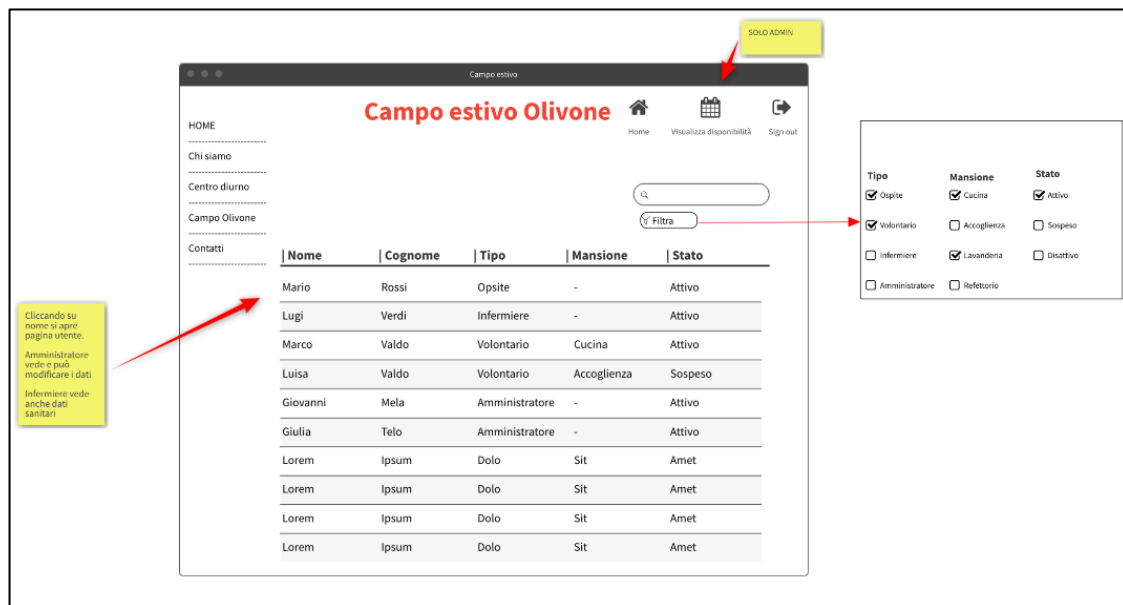


Figura 8 Pagina esplorativa

### 3.3.2 Interfacce di livello 2

In questo capitolo sono raccolte le interfacce visibili solo ad un infermiere.

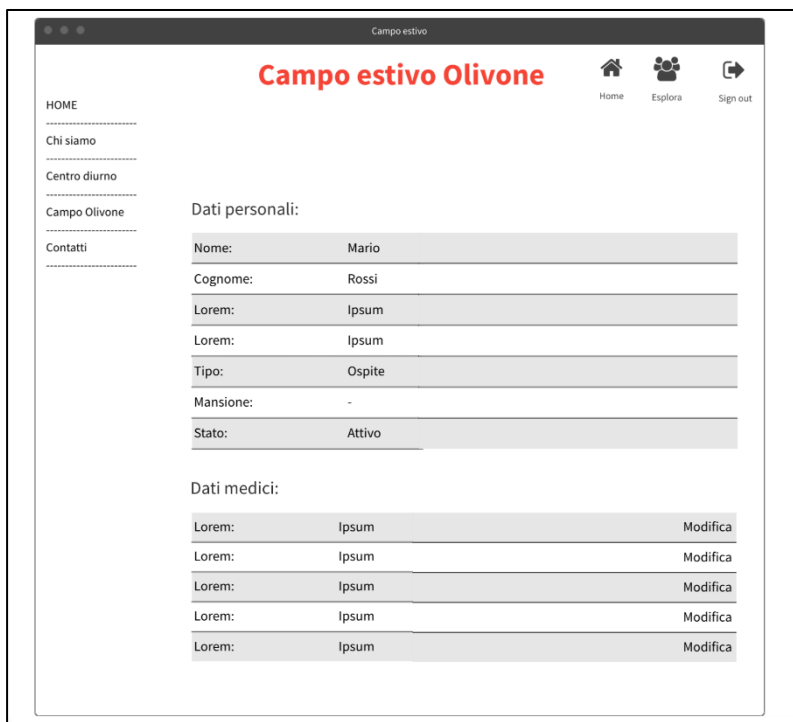


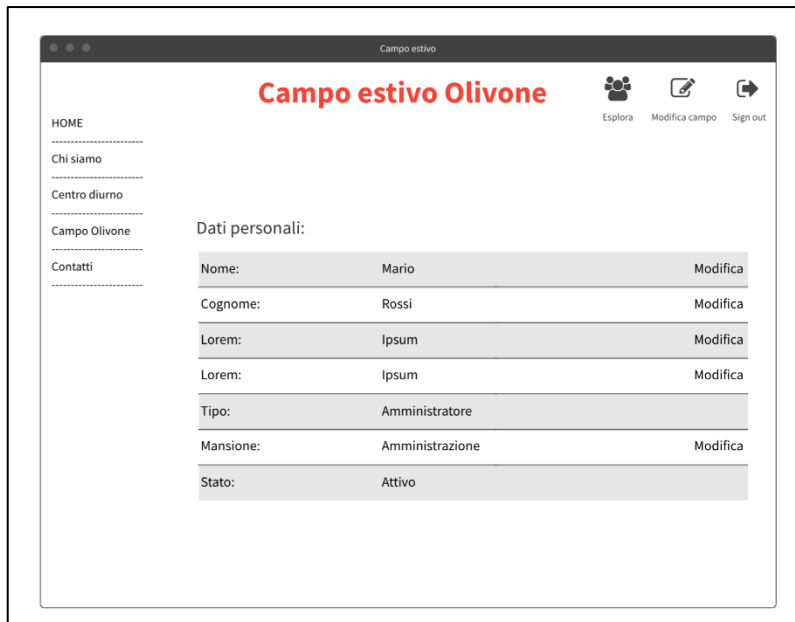
Figura 9 Pagina utente per infermiere

Come detto l'infermiere, oltre ai dati anagrafici, può vedere anche i dati sanitari. Questo è un esempio di come verranno mostrati i dati.

L'utente può muoversi nel sito usando la barra superiore.

### 3.3.3 Interfacce di livello3

In questo capitolo sono raccolte le interfacce visibili solo ad un amministratore.



Dati personali:		
Nome:	Mario	Modifica
Cognome:	Rossi	Modifica
Lorem:	Ipsum	Modifica
Lorem:	Ipsum	Modifica
Tipo:	Amministratore	
Mansione:	Amministrazione	Modifica
Stato:	Attivo	

Questa è la pagina personale di un amministratore. Rispetto ad un utente normale l'admin ha un accesso quasi completo ai suoi campi: lo stato è bloccato per motivi di sicurezza, così come il tipo.

Inoltre con la barra superiore l'amministratore può accedere alla modifica dei dati del campo.

Figura 10 Pagina utente personale admin

Questa è la pagina appena menzionata. Qui l'amministratore può modificare le informazioni del campo.

Cliccando il pulsante "modifica", potrà modificare il testo direttamente sulla pagina.

Potrà inoltre caricare una nuova *brochure*, che andrà a sovrascrivere quella già presente.



**DATE:**  
25 giugno 2022 - 8 Luglio 2022

**Informazioni sul campo:**  
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Vitae turpis massa sed elementum tempus. Condimentum id venenatis a condimentum vitae sapien. In hac habitasse platea dictumst vestibulum. Potenti nullam ac tortor vitae purus faucibus ornare suspendisse. Nec dui nunc mattis enim ut tellus elementum sagittis. Eu mi bibendum neque egestas congue. Maecenas accumsan lacus vel facilisis volutpat est velit egestas. Rutrum quisque non tellus orci ac auctor augue mauris augue. Convallis a cras semper auctor.

Modifica la brochure

Modifica

Salva

Figura 11 Pagina modifica campo

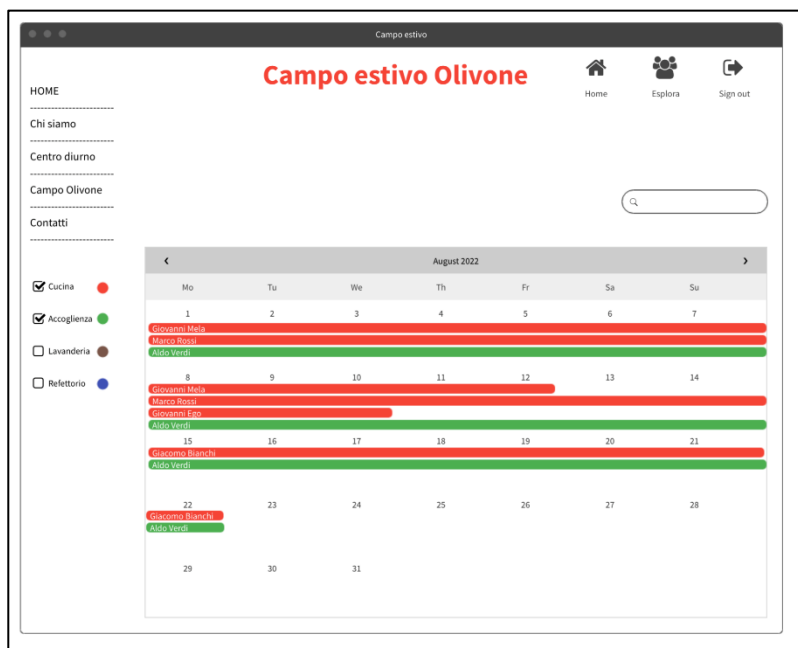


Figura 12 Pagina disponibilità

La pagina di disponibilità raggiungibile dalla pagina di esplorazione (Figura 8), permette all'admin di avere una visione di insieme delle disponibilità dei volontari.

Tramite i filtri a lato del calendario, è possibile visualizzare uno o più gruppi di volontari. È stato pensato per facilitare il compito all'amministratore che deve gestire quanti e quali persone sono assegnate alle varie mansioni.

Questo è un esempio di come un amministratore vede la pagina di un altro utente. Come si vede ha potere di modifica su tutti i campi, ma non vede i dati medici.

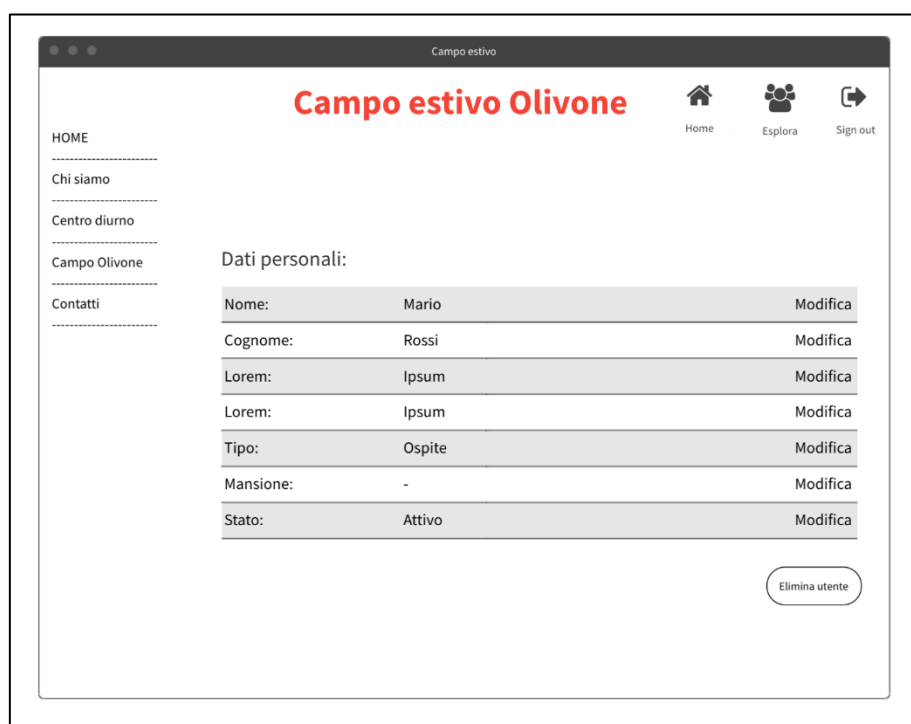


Figura 13 Pagina utente amministratore

### 3.4 Design procedurale

Per questo capitolo ho voluto creare degli schemi di flusso. Uno solo non sarebbe bastato, infatti il campo prevede, come detto nei capitoli precedenti, l'accesso a più tipi di utenti: questo prevede diversi comportamenti da parte del sito.

Di seguito riporto gli schemi, dando una breve spiegazione.

#### 3.4.1 Diagramma di flusso ospite

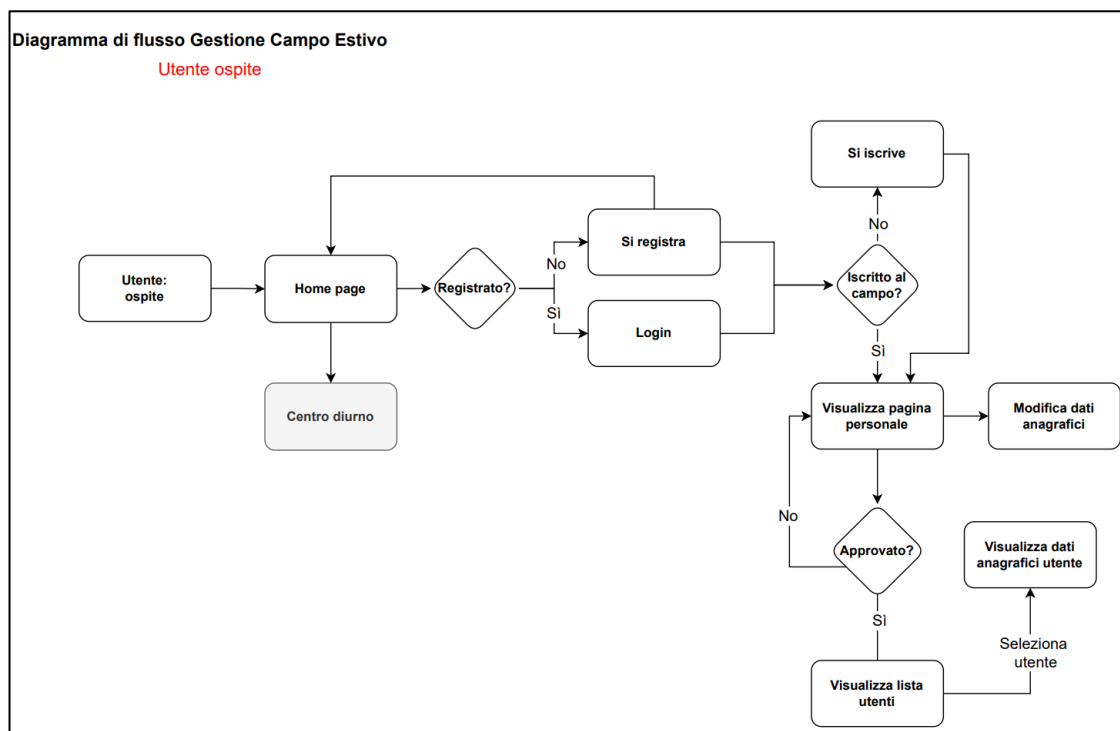


Figura 14 Diagramma di flusso ospiti

Il primo diagramma di flusso è quello che descrive l'uso del sito da parte di un ospite della fondazione. Una volta raggiunta la **home page del campo estivo**, e non quella del sito in generale, l'ospite ha la possibilità di registrarsi o di effettuare il *login*.

Quando si sarà autenticato, se non lo ha già fatto, si può iscrivere al campo estivo presente sul sito. Poi viene reindirizzato alla sua pagina personale, dove può vedere e modificare i suoi dati.

Per ragioni di sicurezza il suo account rimane in sospeso, momentaneamente disattivato. Un amministratore, una volta eseguiti i controlli e gli accertamenti del caso, procederà alla sua attivazione. Ho pensato di implementare questo blocco poiché una volta approvato, l'account può visualizzare i dati anagrafici degli utenti del campo, perciò è opportuno che gli account vengano vidimati prima di avere questo accesso, per evitare utenti indesiderati con intenzioni maligne.

Quando l'account è stato approvato l'ospite può accedere alla lista degli utenti del campo, e visualizzarne i dati anagrafici.

### 3.4.2 Diagramma di flusso volontari ed infermieri

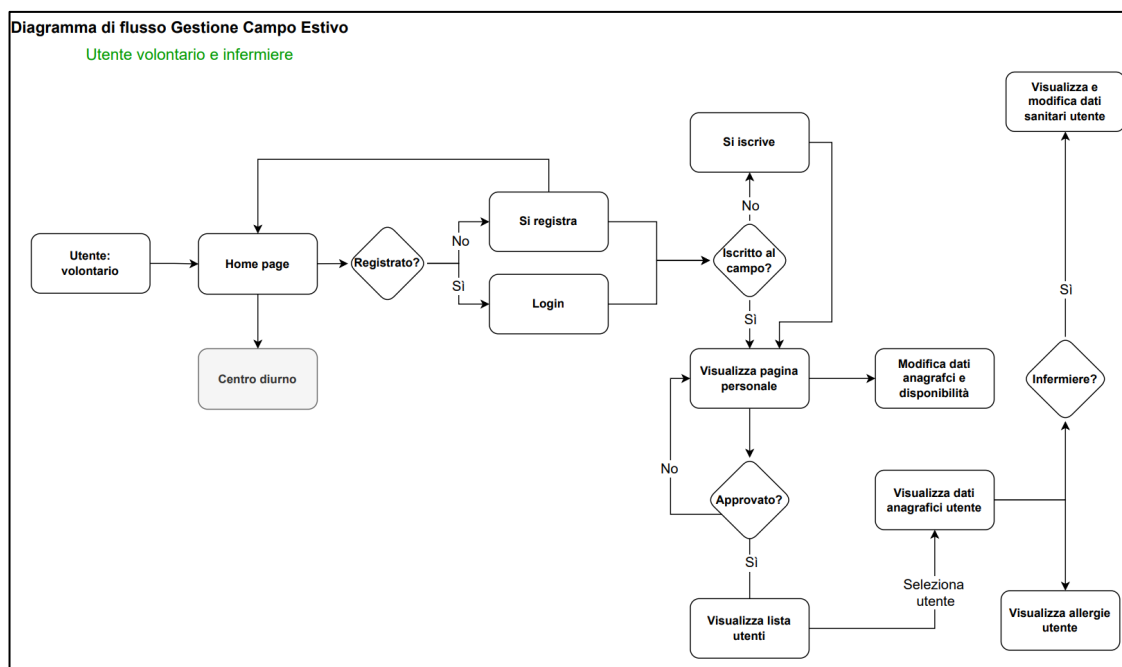


Figura 15 Diagramma di flusso volontari e infermieri

Benché differenti sotto alcuni aspetti, i volontari e gli infermieri sono la stessa cosa a livello organizzativo nel campo. Infatti un infermiere è un volontario approvato dall'amministratore e dal medico del campo.

La procedura di iscrizione al sito e al campo è uguale a quella descritta per l'ospite, così come il blocco sull'account appena creato. In questo caso il blocco acquisisce importanza, poiché un account registrato come volontario può vedere solo le allergie, mentre un account infermiere può, in aggiunta, visualizzare e modificare i dati medici.

Infatti come illustrato anche dallo schema, una volta raggiunta la pagina di un utente del campo, in base al tipo di account, vengono mostrati anche i dati più sensibili degli utenti.

### 3.4.3 Diagramma di flusso amministratori

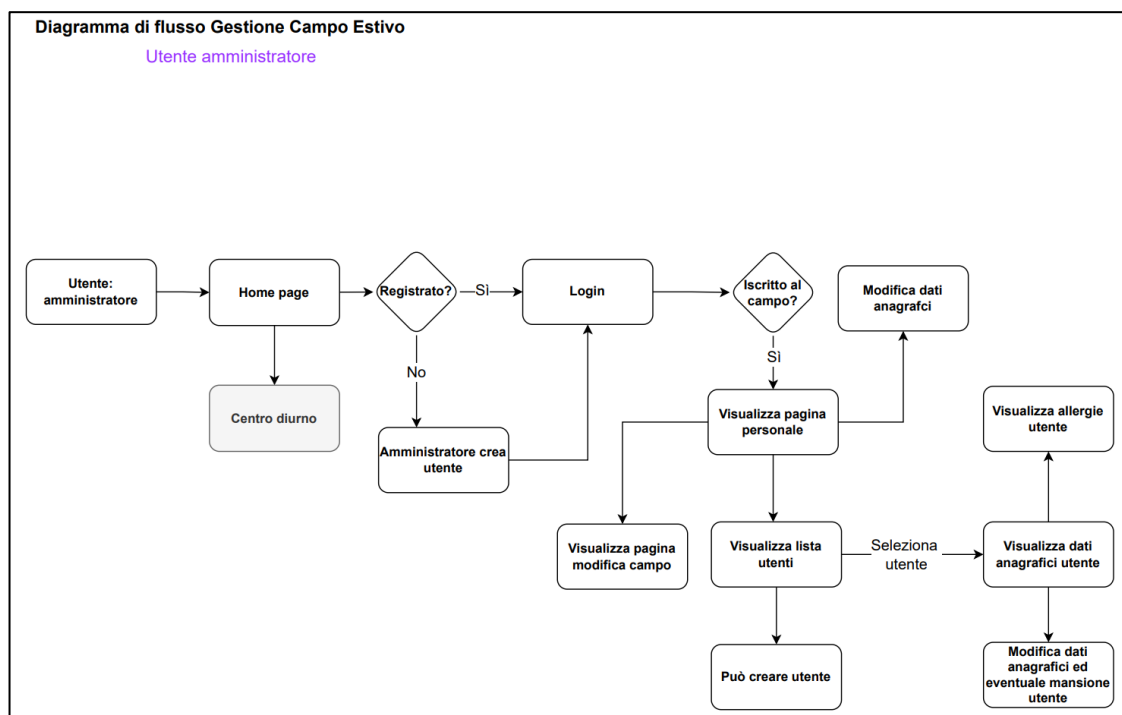


Figura 16 Diagramma di flusso amministratori

L'amministratore è l'account più potente del sito, può fare pressoché tutto, ad esclusione di visualizzare i dati medici.

La procedura di *login* è sempre la stessa, ma un utente non può, logicamente, crearsi un account amministratore in autonomia: solo un altro amministratore può farlo.

Per *default* tutti gli amministratori sono iscritti al campo, in questo modo la gestione dello stesso si semplifica.

Una volta raggiunta la pagina personale, l'amministratore può modificare le informazioni e la descrizione del campo attuale. Può, come tutti, visualizzare la lista di utenti e crearne uno nuovo.

Inoltre può, dopo aver selezionato un utente, visualizzarne e modificarne tutti i dati, eccetto quelli medici.



## 4 Implementazione

### 4.1 Il Database

Per il mio database ho optato per la scelta più classica ma a mio avviso più adatta al progetto, un DB relazionale con MySQL. Per la realizzazione sono ovviamente partito dallo schema ER, nello specifico dalla seconda versione, che era una rappresentazione quasi perfetta di quello che sarebbe stato poi implementato.

#### 4.1.1 Tabella Person

La prima tabella che ho realizzato è la tabella *person*. In questa tabella verranno memorizzati tutti gli utenti del sito. Sono presenti attributi comuni a tutti quanti, come il nome, il cognome, la data di nascita, ecc. Questa tabella contiene anche un *id* auto incrementale che è la chiave per tutti gli utenti. Infine c'è un campo *enum* che specifica il tipo di utente: *guest*, *volontario*, *admin*, *reference*.

- Guest è l'ospite del campo
- Volontario è un volontario del campo
- Admin è l'amministratore
- Reference è la persona di riferimento di un'altra

#### 4.1.2 Entità legate a ospiti

Tutti gli ospiti hanno poi delle entità legate. Infatti, come ho detto nella progettazione, c'erano molti campi da rappresentare, per questo ho diviso tutto in tabelle secondarie. Di seguito un esempio di tabella che sarà popolato da dati riferiti ad un *guest*. Si può vedere come questa tabella abbia un altro *id* che viene usato come chiave e una *foreign key* che permette di far riferimento all'ospite nella tabella *person*.

La maggior parte dei campi sono *NOT NULL*, vorrà dire che andranno per forza riempiti con qualcosa. Non è così per tutti i campi però, infatti *instituteNumber* accetta anche valori nulli. Questo perché suddetto campo è da riempire solo se viene selezionata l'opzione "in istituto" nel campo precedente. Questo non è tuttavia un controllo che può essere fatto a livello di SQL, ma andrà fatto con l'implementazione della pagina.

```
CREATE TABLE livingSituation(
  id INT(20) AUTO_INCREMENT NOT NULL,
  idGuest INT(20) NOT NULL,

  livingState ENUM('da solo', 'con il coniuge', 'in istituto', 'con i parenti') NOT NULL,
  instituteNumber INT(20),

  PRIMARY KEY(id),
  FOREIGN KEY (idGuest) REFERENCES person(id)
  ON UPDATE NO ACTION
  ON DELETE CASCADE
);
```

Le entità collegate all'ospite sono:

- guestInfo
- referencePerson
- livingSituation
- healthIns
- movement
- lastRecover
- alimentation
- guestHyg
- communication
- sleep
- psySate

#### 4.1.3 Entità legate a volontari

Il meccanismo di funzionamento per le entità dei volontari è in tutto e per tutto uguale a quello in uso per le entità *guest*. Unica menzione speciale è l'entità *reference*. Qui sono memorizzati i collegamenti tra persona di riferimento e ospite. Per farlo ho utilizzato una doppia FK, in questo modo:

```

DROP TABLE IF EXISTS referencePerson;
CREATE TABLE referencePerson(
  id INT(20) AUTO_INCREMENT NOT NULL,
  idGuest INT(20) NOT NULL,
  idReference INT(20) NOT NULL,

  PRIMARY KEY(id),
  FOREIGN KEY (idGuest) REFERENCES person(id),
  FOREIGN KEY (idReference) REFERENCES person(id)
  ON UPDATE NO ACTION
  ON DELETE CASCADE
);

```

Vediamo comunque che tutte le entità hanno un *id* che utilizzano come chiave.

Le entità legate al volontario sono:

- referencePerson
- autocertificate

#### 4.1.4 Entità legate ad amministratore

Per questa entità ho usato nuovamente lo stesso meccanismo, la tabella è volta a rappresentare il collegamento tra un amministratore e un campo. Viene usata in fase di creazione e modifica del campo stesso, in modo da avere uno storico dei campi e delle modifiche.


```

CREATE TABLE edit(
  id INT(20) AUTO_INCREMENT NOT NULL,

  last_mod DATE NOT NULL,
  idAdmin INT(11) NOT NULL,
  idCamp INT(11) NOT NULL,

  PRIMARY KEY(id),
  FOREIGN KEY (idAdmin) REFERENCES person(id),
  FOREIGN KEY (idCamp) REFERENCES camp(id)
);

```

	<b>SAMT – Sezione Informatica</b>	Pagina 27 di 47
	<b>Gestione Campo Estivo</b>	

#### 4.1.5 Entità generiche

Infine l'ultima entità del DB, che non è propriamente legata ad un utente, è quella che memorizza i campi estivi. Qui sono presenti dati come il titolo del campo, le novità e le informazioni, le direttive al personale ecc.

## 4.2 Sito web

Come primo passo per la realizzazione di questo progetto ho deciso di realizzare le pagine HTML, non partendo da uno sviluppo PHP bensì da uno puramente grafico. Questo mi ha permesso di avere da subito un'idea di come la pagina avrebbe funzionato alla fine; inoltre, in questo modo ho identificato alcune falle nel mio design, come le icone sbagliate o i collegamenti non ottimali, che ho potuto correggere subito e facilmente, visto che non avevo codice complesso da gestire. Per realizzare le pagine ho deciso di utilizzare il CMS, Nicepage. Un CMS, acronimo di **C**ontent **M**anagement **S**ystem, è un software che permette la creazione di pagine utilizzando l'approccio grafico. Trascinare i componenti, cambiando lo stile, e creare la pagina in questo modo, mi ha permesso di creare il sito in poco tempo, pur facendo un discreto lavoro.

### 4.2.1 Creazione delle views

Per la creazione delle *views*, come detto prima, ho utilizzato un CMS. Non mi dilungherò troppo nella spiegazione poiché non c'è molto da documentare; le *views* sono delle semplici pagine HTML con un CSS allegato.

Una cosa particolare di Nicepage e delle pagine da me create, è il fatto che è previsto l'utilizzo di un *header* sempre uguale. Per la pagina aperta al pubblico ho quindi creato un semplice *header* con il collegamento alle pagine:

- Home
- Contatti
- Chi siamo

Tuttavia questo non poteva funzionare per il resto delle mie pagine. Infatti la maggior parte delle pagine interne del sito, quelle che permettono di iscriversi o di gestire il campo estivo, hanno un *header* sempre dinamico. Ho quindi sfruttato un'opzione del CMS che permette di "nascondere l'*header*". Per la maggior parte delle pagine l'*header* non è quindi presente: ho creato a mano un'altra porzione di sito che fa le veci dell'*header*.

## 4.3 Laravel

### 4.3.1 Setup del progetto Laravel

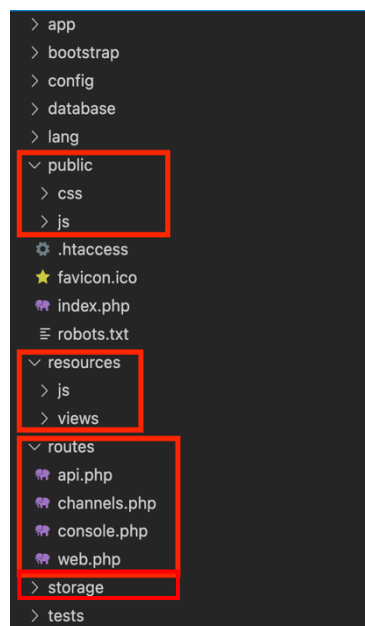
Prima di creare il progetto Laravel, è necessario assicurarsi che sul proprio computer locale siano installati PHP e Composer. Se si sviluppa su MacOS, PHP e Composer possono essere installati tramite Homebrew.

Fortunatamente sul PC fornito dalla scuola entrambi i software erano presenti.  
Per creare il progetto è sufficiente eseguire questo comando

```
composer create-project laravel/laravel example-app
```

Dopo aver creato il progetto è possibile eseguirlo con il comando:

```
php artisan serve
```



Il progetto finito avrà una struttura simile, le sezioni più importanti sono evidenziate.

Nella cartella *public* verranno inseriti tutti i file CSS e JS.

In *resources* c'è una seconda cartella JS, che non viene utilizzata, e la cartella contenente tutte le view del progetto.

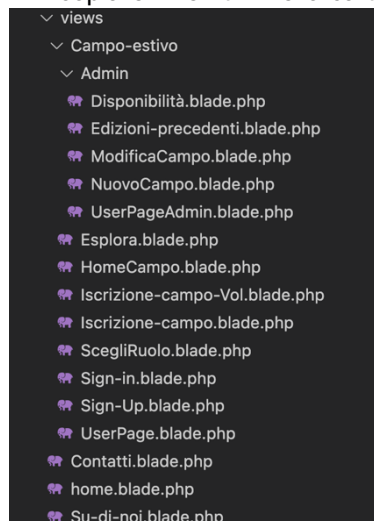
Infine nella cartella *routes* sono contenute tutte le informazioni per il buon funzionamento e la gestione del progetto; come il nome del sito, la versione di Laravel, ecc.

Il file *web.php* è molto importante poiché serve a “mappare” il sito. Tutte le pagine vengono indicate al suo interno, così che Laravel sappia dove cercare quando viene fatto un reindirizzamento.

Infine la cartella *storage* è volta al salvataggio delle immagini del sito.

### 4.3.2 Trasferimento delle views

Per trasferire le views non è bastato fare un semplice copia-incolla. Il primo passo è stato effettivamente copiare i file html nella cartella *views*, i file CSS in quella *CSS*, e così via.



Poi ho dovuto rinominare tutte le *views* seguendo una specifica nomenclatura. Infatti Laravel per identificare una *view* usa una specie di *template*. Per esempio: il file [home.html](#) è diventato [home.blade.php](#). Oltre a convertire il file da HTML a PHP ho quindi aggiunto la *keyword* *blade*.

Nella foto si possono vedere tutte le *views* del progetto.

Un altro accorgimento che ho dovuto prendere è stato nell'adattare i collegamenti tra le pagine. Infatti il classico approccio che prevede l'uso di *href* in questo caso non funziona. Ho dovuto, nel file *web.php*, inserire tutte le pagine con la posizione ed un nome, di seguito un esempio.

Figura 17 Struttura sito

```
Route::get('/Campo-estivo/Iscrizione-campo', function () {
    return view('Campo-estivo/Iscrizione-campo');
})->name('Iscrizione-campo')->middleware('auth');
```

Si utilizza quindi una funzione per ottenere il riferimento passando il percorso assoluto del file partendo dalla cartella *views*. In seguito si può specificare un nome tramite il quale viene identificata la pagina.

Il costrutto *middleware* viene utilizzato per specificare che l'accesso a questa pagina può avvenire solo se l'utente è autenticato al sito. Per gestire l'autenticazione ho usato un'estensione di Laravel chiamata Fortify. Ne parlo nel prossimo capitolo.

Per utilizzare questa route appena creata è sufficiente, in un attributo HTML che lo permette, includere la seguente istruzione:

```
href="{{route('Iscrizione-campo')}}"
```

Specificando il nome scelto nel file *web.php* è possibile raggiungere comodamente qualunque pagina.

Non sono solo i link delle pagine ad essere cambiati, ma anche quelli dei file CSS.

In questo caso però non è necessario indicarne la posizione nel file *web.php*, infatti Laravel prevede che essi si trovino tutti nella cartella apposita citata in precedenza. Per questo è sufficiente fare:

```
<link rel="stylesheet" href="{{ URL::asset('css/Contatti.css') }}" />
```

Stesso meccanismo viene utilizzato per le immagini, ecco un esempio:

```

```

### 4.3.3 Login in Laravel

Per gestire il Login, come ho utilizzato Fortify.

Come accennato in precedenza Fortify è un'implementazione *backend* di autenticazione per Laravel. Fortify implementa le routes e i controller necessari per implementare tutte le funzioni di autenticazione di Laravel, tra cui il login, la registrazione, la reimpostazione della password, la verifica dell'e-mail, ecc.

Non è obbligatorio usare Fortify per utilizzare le funzioni di autenticazione di Laravel. Potevo infatti implementare manualmente tutte queste funzioni, tuttavia è consigliato utilizzare questo framework poiché è tra i più stabili.

Per installarlo è sufficiente eseguire i seguenti comandi:

```
composer require laravel/fortify
php artisan vendor:publish --provider="Laravel\Fortify\FortifyServiceProvider"
php artisan migrate
```

L'ultimo comando serve a inserire nel DB le tabelle che Fortify utilizza per memorizzare gli utenti. Un vantaggio che si ha seguendo questo approccio è che la password è cifrata di default.

Poi dobbiamo includere manualmente l'*app service provider* nel progetto, un po' come fare l'import di una libreria in un normale codice sorgente. Per farlo bisogna aprire il file `/config/app.php`, qui bisogna inserire il collegamento a Fortify.

Nella sezione *providers* incollare quindi:

```
App\Providers\FortifyServiceProvider::class,
```

La configurazione iniziale non è finita, nel file `/providers/AppServiceProvider.php` bisogna indicare che *view* devono venir utilizzate per il login e per la registrazione. Qui troveremo un metodo *boot* che è di default, al suo interno inserire le funzioni per specificare le *views*.

```
public function boot(){
    Fortify::loginView(function(){
        return view("Campo-estivo/Sign-In");
    });
    Fortify::registerView(function(){
        return view("Campo-estivo/Sign-Up");
    });
}
```

Ora si può finalmente utilizzare il login, assicurandosi che in entrambe le view specificate sia presente un *form* che abbia i corretti parametri dichiarati, ad esempio il metodo POST e la route corretta.

Inoltre è fondamentale che anche i campi per l'email e la password rispettino anch'essi una specifica sintassi. Ecco il mio form di registrazione, che oltre al nome e all'email, necessari per il buon funzionamento del form, presenta pure un campo per la conferma della password. L'ho semplificato togliendo lo stile ed altre cose superflue al contesto di Fortify.

```
<form method="POST" action="{{route('register')}}"> @csrf

    <!-- User name-->
    <label for="name">{{__('Name')}}</label>
    <input id="name" name="name" value="{{old('name')}}" required>
    @error('name')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror

    <!-- User e-mail -->
    <label for="email">{{__('E-Mail Address')}}</label>
    <input id="email" type="email" name="email" value="{{old('email')}}" required>
    @error('email')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror

    <!-- User password -->
    <label for="password">{{__('Password')}}</label>
    <input id="password" type="password" name="password" required>
    @error('password')
        <span class="invalid-feedback" role="alert">
            <strong>{{ $message }}</strong>
        </span>
    @enderror

    <!-- Password confirm-->
    <label for="password-confirm">{{__('Confirm Password')}}</label>
    <input id="password-confirm" type="password" name="password_confirmation" required>

    <!-- Button send-->
    <button type="submit">
        {{ __('Register') }}
    </button>
</form>
```

Il form di login è pressoché uguale, sia per struttura che sintassi dei campi. Tuttavia, come è logico che sia, ha solo 2 campi: uno per l'email e uno per la password. L'unica differenza degna di nota sta nella prima riga, dove viene specificato il tipo di form e la route. In questo caso la route è *login*.

```
<form method="POST" action="{{route('login')}}">
    ...
</form>
```

## 4.4 Realizzazione del progetto

### 4.4.1 Home principale

Nella home principale, seguendo i requisiti, ho incluso due semplici link che permettono di collegarsi rispettivamente al sito della fondazione e alla home page del campo. Di come funziona un link ho già parlato precedentemente, il meccanismo qui è lo stesso.

### 4.4.2 Home campo

Questa è la pagina principale del campo: qui sono presenti i collegamenti alle pagine di *Sign up* e di *login*. Tramite i link in cima alla pagina si può inoltre accedere alla propria pagina utente e iscriversi al campo. In questa foto, che rappresenta una sezione della home page, non sono visualizzabili le icone, ciò è dovuto a un problema presente sul mio portatile e non del progetto infatti esse sono visualizzabili sul sito della scuola.



### 4.4.3 Registrazione e login

Di questo aspetto ho già parlato in modo esaustivo nel capitolo dedicato all'autenticazione in Fortify. Anche in questo caso quindi non mi dilungherò dato che non ho implementato nulla di più rispetto a quanto già documentato.

Unico aspetto degno di nota è che inizialmente avevo previsto che il login e la registrazione avvenissero tramite un pop-up. Tuttavia alla fine ho utilizzato una pagina poiché Fortify lo prevedeva, mi sono quindi dovuto adattare.


### 4.4.4 Registrazione al campo



Prima di registrarsi ad un campo è necessario scegliere il proprio ruolo, che può essere volontario o ospite.

Questo viene fatto tramite una pagina intermedia una volta che si clicca su "iscriviti al campo".



	<b>SAMT – Sezione Informatica</b>	Pagina 33 di 47
	<b>Gestione Campo Estivo</b>	

Una volta scelto il ruolo si viene presentati con un form che permette l'iscrizione al campo.

Per creare il form mi sono basato su quello che ho già fatto per il *login*.  
Di seguito la dichiarazione del form.

```
<form action="{{route('QueryAllData')}}" method="POST" name="form"> @csrf
```

Si può vedere che nell'action viene specificato *QueryAllData*.  
Questo non è altro che un metodo da me creato nel controller della pagina. Con il seguente comando ho creato un controller:

```
php artisan make:controller NomeController --invokable
```

Poi nel file *web.php* ho definito la route che collegasse il form nella pagina di iscrizione alla funzione *QueryAllData*.

```
Route::post("/Campo-estivo/Iscrizione-campo",
[\App\Http\Controllers\IscrizioneOspController::class, 'QueryAllData'])->name('QueryAllData');
```

Questo mi ha permesso di passare allo sviluppo della funzione vera e propria:

#### 4.4.4.1 QueryAllData

Il primo passo è stato prelevare i dati dal form, per farlo ho usato la seguente sintassi:

```
public function QueryAllData(Request $request){
    $nameTxt = $request->nameTxt;

    ...
}
```

Dichiarando *\$request* posso prendere i dati semplicemente indicando il nome del campo al quale voglio accedere.

Ho ripetuto questo processo per tutti i campi del form, poi sono passato al salvataggio nel DB.

Per farlo ho usato molto la guida ufficiale di Laravel e ho trovato un metodo semplice per farlo, ecco un esempio:

```
/*
 * Inserimento di un set di dati in una tabella.
 */
DB::table("person")->insert([
    "name" => $nameTxt,
    "lastname" => $lastNameTxt,
    "phone_number" => $phoneNumber,
    "address" => $viaTxt,
    "CAP" => $CAP,
    "country" => $nazione,
    "type" => "guest",
    "born_date" => $bornDate,
]);
```

Specificando la tabella e usando il costrutto *insert* posso inserire i dati presi dai campi direttamente nel DB.

Questo esempio rappresenta il primo inserimento che ho fatto. Poi, con il codice seguente, ho salvato l'id dell'utente appena inserito in modo da avere la FK da utilizzare per tutti gli altri inserimenti:

```
$idGuest = DB::table("person")
    ->latest("id")
    ->first();
$idGuest = $idGuest->id;
```

Per il resto delle tabelle ho usato lo stesso procedimento usato per la tabella *person*.

Dopo aver inserito tutti i dati ho ritornato la *view* della pagina utente.

```
return View::make('Campo-estivo/UserPage');
```

L'utente, dopo la registrazione, viene così reindirizzato alla sua pagina personale.

#### 4.4.5 User page

Per la pagina utente ho utilizzato un approccio leggermente diverso rispetto a prima. Ho infatti realizzato tutto il codice PHP direttamente nella pagina, senza utilizzare un controller. Il mio obiettivo era ottenere una lista degli attributi dell'utente corrente. A questo scopo ho usato la classe *Auth*, implementata nativamente in Laravel/Fortify.

```
$user=Auth::user()->name;
```

Ho poi selezionato l'utente dal DB e l'ho salvato in una variabile

```
$results = DB::select("select * from person where name = '".$user.'");
```

Poi ho definito un array che contenesse le etichette di tutti gli attributi dell'utente:


```
$data = array("Id", "Nome", "Cognome", "Numero di telefono",
    "Via", "CAP", "Nazione", "Ruolo", "Data di nascita", "");
```

Infine ho scritto il ciclo che stampasse tutti i dati:

Ho dovuto fare un doppio ciclo poiché dal database viene estratta una matrice. Perciò il primo ciclo itera tutte le "righe" della matrice, ogni riga corrisponde ad un dato dell'utente, quindi: nome, cognome, ecc. Il secondo ciclo itera attraverso l'array temporaneo appena creato e mi permette di concatenare così l'etichetta e il dato dell'utente.

Ho dovuto usare la funzione `array_values` poiché i dati estratti non usavano un indice numerico ma una stringa.

```
foreach ($results as $el){
    $realArray = (array)$el;
    $realArray=array_values($realArray);
    for ($i=0;$i<count($realArray);$i++){
        print("<tr><td>".$data[$i]."</td><td>".$realArray[$i]."</td></tr>");
    }
}
```

	<b>SAMT – Sezione Informatica</b>	Pagina 35 di 47
	<b>Gestione Campo Estivo</b>	

Questo è tutto quello che sono riuscito a fare per quanto riguarda l'implementazione: La scarsità di foto e screen dipende dalla mia impossibilità ad eseguire correttamente il mio progetto in remoto.

## 5 Test

### 5.1 Protocollo di test

Test Case:	TC-01	Nome: Collegamento al sito della fondazione
Riferimento:	REQ-01	
Descrizione:	La pagina base ha il collegamento al sito della fondazione	
Prerequisiti:	-	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Centro diurno”</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si viene reindirizzati al sito della fondazione</li> </ol>	

Test Case:	TC-02	Nome: Registrazione al sito
Riferimento:	REQ-02	
Descrizione:	È possibile registrarsi al sito	
Prerequisiti:	-	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Cliccare su “Sign-up” e registrarsi</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si viene correttamente registrati</li> </ol>	

Test Case:	TC-03	Nome: Login al sito
Riferimento:	REQ-03	
Descrizione:	È possibile autenticati al sito	
Prerequisiti:	-	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Cliccare su “Sign-in” e registrarsi</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si viene correttamente autenticati</li> </ol>	

Test Case:	TC-04	Nome: Iscrizione al campo
Riferimento:	REQ-04	
Descrizione:	È possibile iscriversi al campo	
Prerequisiti:	-	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “iscrizione al campo”</li> <li>5. Inserire tutti i dati</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si viene correttamente registrati al campo</li> </ol>	

Test Case:	TC-05	Nome: User page ospite
Riferimento:	REQ-05	
Descrizione:	Visualizzare i dati del campo	
Prerequisiti:	Un ospite registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Controllare la presenza di dati e modificarne uno</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. I dati sono consultabili e modificabili</li> </ol>	

Test Case:	TC-06	Nome: Pagine esplorative ospite
Riferimento:	REQ-06	
Descrizione:	Un ospite loggato può visualizzare i dati di altri utenti	
Prerequisiti:	Un ospite registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si visualizza un’anteprima dei dati anagrafici di altri utenti</li> </ol>	

Test Case:	TC-07	Nome: Pagine esplorative filtrabili
Riferimento:	REQ-06	
Descrizione:	Un utente loggato può visualizzare i dati di altri utenti e ordinarli	
Prerequisiti:	Un utente registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a filtrare gli utenti cliccando sul titolo della lista</li> </ol>	
Risultati attesi:	1. Si visualizza un’anteprima dei dati anagrafici di altri utenti	

Test Case:	TC-08	Nome: Pagine personali altri utenti
Riferimento:	REQ-06	
Descrizione:	Un utente loggato può visualizzare i dati di un utente aprendolo dalla pagina esplorativa	
Prerequisiti:	Un utente registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a cliccare su un utente</li> </ol>	
Risultati attesi:	1. Si visualizzano tutti i dati anagrafici di un utente	

Test Case:	TC-09	Nome: Pagine personali altri utenti
Riferimento:	REQ-07	
Descrizione:	Un utente loggato può visualizzare i dati di un utente aprendolo dalla pagina esplorativa	
Prerequisiti:	Un utente registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a cliccare su un utente</li> </ol>	
Risultati attesi:	1. Si visualizzano tutti i dati anagrafici di un utente	

Test Case:	TC-10	Nome: Pagine esplorative volontario
Riferimento:	REQ-08	
Descrizione:	Un volontario loggato può visualizzare i dati di altri utenti	
Prerequisiti:	Un volontario registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si visualizza un’anteprima dei dati anagrafici di altri utenti</li> </ol>	

Test Case:	TC-11	Nome: Pagine esplorative filtrabili
Riferimento:	REQ-08	
Descrizione:	Un volontario loggato può visualizzare i dati di altri utenti e ordinarli	
Prerequisiti:	Un volontario registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a filtrare gli utenti cliccando sul titolo della lista</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si visualizza un’anteprima dei dati anagrafici di altri utenti</li> </ol>	

Test Case:	TC-12	Nome: Pagine personali altri utenti
Riferimento:	REQ-08	
Descrizione:	Un volontario loggato può visualizzare i dati di un utente aprendolo dalla pagina esplorativa	
Prerequisiti:	Un volontario registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a cliccare su un utente</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Si visualizzano tutti i dati anagrafici di un utente e le sue allergie</li> </ol>	

Test Case:	TC-13	Nome: Pagine personali altri utenti
Riferimento:	REQ-08	
Descrizione:	Un volontario loggato può visualizzare i dati di un utente aprendolo dalla pagina esplorativa	
Prerequisiti:	Un volontario registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a cliccare su un utente</li> </ol>	
Risultati attesi:	1. Si visualizzano tutti i dati anagrafici di un utente	

Test Case:	TC-14	Nome: Pagine personali altri utenti infermiere
Riferimento:	REQ-08	
Descrizione:	Un infermiere loggato può visualizzare i dati di un utente aprendolo dalla pagina esplorativa	
Prerequisiti:	Un infermiere registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a cliccare su un utente</li> </ol>	
Risultati attesi:	1. Si visualizzano tutti i dati anagrafici di un utente e i suoi dati medici	

Test Case:	TC-15	Nome: User page admin
Riferimento:	REQ-09	
Descrizione:	Visualizzare i dati del campo	
Prerequisiti:	Un admin	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Controllare la presenza di dati e modificarne uno</li> </ol>	
Risultati attesi:	1. I dati sono consultabili e modificabili	



Test Case:	TC-16	Nome: Pagine esplorative admin
Riferimento:	REQ-10	
Descrizione:	Un admin loggato può visualizzare i dati di altri utenti	
Prerequisiti:	Un admin registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> </ol>	
Risultati attesi:	1. Si visualizza un’anteprima dei dati anagrafici di altri utenti	

Test Case:	TC-17	Nome: Pagine esplorative filtrabili admin
Riferimento:	REQ-10	
Descrizione:	Un admin loggato può visualizzare i dati di altri utenti e ordinarli	
Prerequisiti:	Un admin registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a filtrare gli utenti cliccando sul titolo della lista</li> </ol>	
Risultati attesi:	1. Si visualizza un’anteprima dei dati anagrafici di altri utenti	

Test Case:	TC-18	Nome: Pagine personali altri utenti admin
Riferimento:	REQ-10	
Descrizione:	Un admin loggato può visualizzare i dati di un utente aprendolo dalla pagina esplorativa. Può inoltre modificare tutti i dati	
Prerequisiti:	Un admin registrato al campo	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Cliccare su “Campo estivo”</li> <li>3. Autenticarsi</li> <li>4. Cliccare su “User page”</li> <li>5. Cliccare su “Esplora”</li> <li>6. Provare a cliccare su un utente</li> </ol>	
Risultati attesi:	1. Si visualizzano tutti i dati di un utente e può modificarli. Non può visualizzare i dati medici	

Test Case:	TC-19	Nome: Creazione e modifica di un campo
Riferimento:	REQ-11	
Descrizione:	Un amministratore, può modificare i dati sul campo corrente o su quelli passati. Inoltre può creare una nuova edizione del campo, inserendo tutte le informazioni necessarie.	
Prerequisiti:	Un admin autenticato al sito	
Procedura:	1. Meccanica non prevista	
Risultati attesi:	1.	

Test Case:	TC-20	Nome: Creazione account ospite
Riferimento:	REQ-12	
Descrizione:	Un amministratore può autonomamente creare un account per un ospite.	
Prerequisiti:	Un admin autenticato al sito	
Procedura:	1. Meccanica non prevista	
Risultati attesi:		

Test Case:	TC-21	Nome: Attivazione ed eliminazione account
Riferimento:	REQ-13	
Descrizione:	Un amministratore può attivare o eliminare l'account di un volontario o di un ospite.	
Prerequisiti:	Un admin autenticato al sito	
Procedura:	1. Meccanica non prevista	
Risultati attesi:		

Test Case:	TC-22	Nome: Disponibilità volontari
Riferimento:	REQ-14	
Descrizione:	Un amministratore può visualizzare la disponibilità dei suoi volontari	
Prerequisiti:	Un admin autenticato al sito	
Procedura:	1. Meccanica non prevista	
Risultati attesi:		

Test Case:	TC-23	Nome: Modifica informazioni campo
Riferimento:	REQ-15	
Descrizione:	Un amministratore può cambiare le informazioni del campo	
Prerequisiti:	Un admin autenticato al sito	
Procedura:	1. Meccanica non prevista	
Risultati attesi:		

Test Case:	TC-24	Nome: Sito responsive
Riferimento:	REQ-15	
Descrizione:	Il sito è <i>responsive</i>	
Prerequisiti:	-	
Procedura:	<ol style="list-style-type: none"> <li>1. Collegarsi al sito</li> <li>2. Provare a ridimensionare la pagina</li> </ol>	
Risultati attesi:	<ol style="list-style-type: none"> <li>1. Le informazioni vengono sempre presentate bene</li> </ol>	

## 5.2 Risultati test

Test Case	Esito	Commenti
TC-01	Passato	
TC-02	Passato	
TC-03	Passato	
TC-04	Passato	
TC-05	Parzialmente passato	Dati solo visualizzabili
TC-06	Fallito	
TC-07	Fallito	
TC-08	Fallito	
TC-09	Fallito	
TC-10	Fallito	
TC-11	Fallito	
TC-12	Fallito	
TC-13	Fallito	
TC-14	Fallito	
TC-15	Fallito	
TC-16	Fallito	
TC-17	Fallito	
TC-18	Fallito	
TC-19	Fallito	
TC-20	Fallito	
TC-21	Fallito	
TC-22	Fallito	
TC-23	Fallito	
TC-24	Passato	

## 5.3 Mancanze/limitazioni conosciute

Come visibile dalla tabella di test questo progetto ha molte mancanze. Tutto, o quasi, ciò che riguarda l'implementazione della gestione del campo in sé è completamente assente. Credo che il motivo sia da ricercare principalmente nelle mie assenze. Nelle ultime settimane ero riuscito a procedere con l'implementazione e con la scrittura del codice PHP, dato che per molti aspetti ho potuto riciclare codice che avevo già scritto. Il progetto in sé non presentava molte altre cose nuove infatti. Una volta imparato il meccanismo di lettura e scrittura dei dati sul database infatti si potevano già risolvere molti problemi.

## 6 Consuntivo

Non sono riuscito a produrre un Gantt consuntivo per la mancanza di mezzi e di mobilità. Posso dire che fino alla progettazione sono stato perfettamente nei tempi se non in anticipo.

Per l'implementazione ho avuto qualche piccolo ritardo iniziale che poi ho chiuso, lavorando ad aspetti che avevo previsto di fare più in là nel tempo ma che si sono rivelati fondamentali.

Verso inizio novembre ho poi avuto i primi rallentamenti dovuti a visite in ospedale e malesseri che non mi hanno permesso di continuare a pieno regime.

Infine la seconda settimana di novembre sono stato operato e da lì non ho più proseguito con l'implementazione e con altre parti del progetto.

Solo le prime settimane di dicembre ho potuto proseguire con la documentazione con l'aiuto di mia madre, che mi ha aiutato a scriverla per la maggior parte.

## 7 Conclusioni

Il mio progetto non è per nulla concluso. Mancano, come detto, moltissimi aspetti fondamentali al funzionamento. Per quello che ho realizzato il progetto ora come ora è pressoché inutilizzabile. Tuttavia credo di aver realizzato una base decente per uno sviluppo futuro. Le pagine sono tutte pronte per essere adattate all'uso con Laravel e PHP più in generale. Tuttavia non so quanto aver realizzato lo scheletro prima di dedicarmi all'implementazione sia stata una buona cosa, anche se rimango convinto che mi abbia permesso di risolvere molti problemi con i requisiti e dubbi riguardo l'implementazione non solo del sito, ma anche del database.

Credo invece che la progettazione sia la parte del progetto realizzata meglio. Mi sono concentrato molto non solo sulle interfacce ma anche sul funzionamento del backend e sulla struttura del DB.


Soprattutto per quest'ultimo sono contento: in 3 anni non avevo ancora realizzato un progetto che necessitasse di un DB relazionale, perciò sono felice che finalmente ho potuto confrontarmi con questo aspetto anche in un progetto. Penso che l'aver realizzato due versioni dello stesso schema, con due convenzioni diverse, mi abbia permesso di mettere in luce varie criticità a cui non avevo pensato dopo aver realizzato la prima versione, quella con le convenzioni "standard" della SAMT.

Tuttavia una buona progettazione non basta per bilanciare la mia implementazione: come ho detto nel capitolo scorso non sono riuscito a implementare buona parte del progetto, e sono dispiaciuto per questo. Purtroppo le mie assenze per malesseri e un intervento mi hanno impedito di procedere come avevo preventivato e sperato.

Passando al lato tecnico sono abbastanza contento di aver affrontato questo progetto. Per me lo sviluppo in PHP era argomento nuovo per la tematica dei progetti e in questi mesi ho avuto l'occasione non solo di arricchire le mie conoscenze in questo linguaggio ma anche di imparare ad utilizzare un framework nuovo: Laravel.

So che nel mondo del lavoro e della produzione è un framework molto utilizzato e affidabile, tramite il quale si possono realizzare progetti molto grandi e importanti. Credo che la curva di apprendimento presentata da Laravel sia considerevole e non sia facile all'inizio raccapezzarsi con tutte le sue particolarità e regole. Tuttavia una volta imparati i meccanismi di base si può iniziare a fare sul serio.

In conclusione non sono molto soddisfatto del progetto, però penso di essermi impegnato, nel limite del possibile, a fare del mio meglio. So che questo progetto è reale e non solo un software creato ad hoc per i progetti semestrali, soprattutto per questo sono dispiaciuto perché questo è un prodotto che ha un reale scopo e ci sono persone che hanno una reale necessità.

	<b>SAMT – Sezione Informatica</b>	Pagina 46 di 47
	<b>Gestione Campo Estivo</b>	

## 7.1 Sviluppi futuri

Non ho previsto molti sviluppi futuri: penso che gli sviluppi saranno banalmente il soddisfare i requisiti che io non sono riuscito a completare. Penso che possa essere importante riprendere questo progetto in futuro, non solo per il fatto che è un prodotto con un reale uso, ma per il fatto che di fondo ci sia del bene nello scopo della realizzazione del progetto. Mi spiego meglio: se completato questo progetto semplificherà la vita a persone che si trovano a confronto con grandi difficoltà tutti i giorni: persone disabili e anziane.

## 8 Bibliografia

---

### 8.1 Sitografia

#### Esempio:

- <https://laravel.com/docs/9.x/fortify>
- <https://laravel.com/docs/9.x/database>
- <https://laravel.com/docs/9.x/authentication>
- <https://laravel.com/docs/9.x/routing>
- <https://laravel.com/docs/9.x/middleware>
- <https://laravel.com/docs/9.x/views>
- <https://laravel.com/docs/9.x/blade>
- <https://laravel.com/docs/9.x/installation>
- <https://laravel.com/docs/9.x/configuration>
- <https://laravel.com/docs/9.x/structure>
- <https://laravel.com/docs/9.x/frontend>
- <https://laravel.com/docs/9.x/starter-kits>
- <https://laravel.com/docs/9.x/controllers>
- <https://laravel.com/docs/9.x/requests>
- <https://laravel.com/docs/9.x/responses>
- <https://laravel.com/docs/9.x/session>
- <https://laravel.com/docs/9.x/logging>
- <https://laravel.com/docs/9.x/database>
- <https://laravel.com/docs/9.x/queries>
- <https://laravel.com/docs/9.x/pagination>
- <https://laravel.com/docs/9.x/migrations>
- <https://laravel.com/docs/9.x/seeding>

## 9 Allegati

---

Elenco degli allegati, esempio:

- Diari di lavoro
- Progettazione e schemi
- QdC
- Prodotto