IMMOBILE17

Immobile 17 è un'agenzia immobiliare situata a L'Aquila. Il sito web consente ai clienti, in maniera integrata, di visionare gli immobili gestiti dall'Agenzia e di prendere appuntamenti per visitarli. Tramite opportuni accorgimenti nel layer View e nel FrontController, il server è già API-ready per quanto riguarda la sezione dedicata agli utenti, pur mancando l'effettiva implementazione della View. La comunicazione client-server avviene tramite RESTful APIs con l'utilizzo di metodi GET e POST.

Gli utenti del sito si dividono in 3 tipologie:

- CLIENTE:

Il cliente non registrato può visionare i singoli immobili e filtrare la ricerca dei suddetti tramite vari parametri(prezzo massimo, grandezza minima, parola chiave, tipologia e tipologia di annuncio, affitto o vendita).
La registrazione e la successiva attivazione dell'account tramite email permettono di prenotare un appuntamento per visionare gli immobili, scegliendo esclusivamente finestre orarie in cui sono disponibili agenti immobiliari liberi. L'utente registrato può inoltre, tramite la propria area personale, visionare un calendario degli appuntamenti presi e i dettagli di ogni appuntamento (immobile e agente immobiliare).

- AGENTE IMMOBILIARE:

Le funzionalità dell'Agente Immobiliare sono le stesse del cliente. Egli non può però prenotare un appuntamento.

- AMMINISTRATORE:

L'amministratore ha a disposizione molteplici funzionalità per la gestione dell'Agenzia Immobiliare. Può visionare, aggiungere, modificare, attivare/disattivare ed eliminare clienti, agenti immobiliari, immobili ed appuntamenti.

GESTIONE DELLE URL E FRONT CONTROLLER

Il Front Controller si occupa si occupa di indirizzare nel modo corretto le richieste http ai metodi e alle classi corretti del package Controller.
Le URL sono strutturate nel seguente modo:

host/Immobile17/api/token=xxxx/controller/metodo/

Dove la risorsa api è opzionale(legata al tipo di interfaccia dell'utente) e la risorsa token è anch'essa opzionale, legata al parametro api. Le risorse

controller e metodo definiscono la classe del package controller e il rispettivo metodo da chiamare.
Nel caso in cui il controller non sia 'Admin', viene passato come ulteriore parametro al metodo del controller un booleano api che identifica il tipo di View da utilizzare.

Analizziamo 3 ulteriori URL per vedere come vengono passati i parametri alla funzione:

1. host/Immobile17/Immobile/visualizzaImmobili
2. host/Immobile17/Immobile/visualizza/IM1
3. host/Immobile17/Immobile/ricerca/ti/Vendita/tp/Monolocale

Nel caso 1 alla funzione visualizzaImmobili del controller Immobile viene passato esclusivamente il parametro api(in questo caso false).
Nel caso 2 alla funzione visualizza del controller Immobile vengono passati come parametri la stringa IM1 e il booleano api(false).
Nel caso 3 vengono passati alla funzione ricerca del controller Immobile un array con due tuple (ti=>Vendita e tp=>Monolocale) e il parametro api(false).

Le URL che chiamano funzioni del controller Admin seguono la stessa logica, ma è assente il parametro api, in quanto non è stato ritenuto conveniente introdurre la possibilità di usufruire delle funzioni dell'Admin da app neppure in futuri upgrade.

GESTIONE DELLE SESSIONI E API RESTFUL
Le sessioni, della durata di 1h, contengono esclusivamente il parametro id, che permette di caricare facilmente l'utente in questione dal DB. Le operazioni di creazione e l'eliminazione delle sessioni, di caricamento dell'oggetto MUtente e il controllo dell'esistenza della sessione sono a carico della classe CSessionManager.
Per quanto riguarda le RESTful APIs, al momento del login, se nell'URL è presente la risorsa api, se il login ha avuto successo viene generato un token. Tale token viene caricato nel database previa criptazione e inviato all'utente. Qualora l'utente si connetta tramite interfaccia mobile e la risorsa token sia presente e venga validata, viene creata una sessione e l'utente risulterà quindi loggato.
Lo stesso meccanismo di creazione e gestione dei token viene utilizzato per la funzione "remember me". Al momento del login, qualora spunti la casella "ricordami", viene creato e inserito previa criptazione nel database un token associato all'Id dell'utente. Tale token viene memorizzato in un cookie della durata di 60 giorni. Ad ogni richiesta HTTP, viene controllata la presenza del cookie e la validità del token. Nel caso il cookie sia valido, viene inizializzata una sessione per l'utente in questione.

DESCRIZIONE PROGETTUALE:
Il backend del sito web è stato sviluppato secondo il pattern MVC, con l'obiettivo di creare un ambiente di sviluppo facilmente manutenibile e aggiornabile.
Le caratteristiche dei vari layer in dettaglio:

- FOUNDATION:

Il layer foundation si compone a sua volta di tre layer, ognuno con una specifica funzione:

- FPersistentManager: classe che funge da "proxy", pur non essendolo formalmente, fra i livelli sottostanti e il resto dell'applicativo. Virtualizza le operazioni del DB restituendo solamente gli oggetti richiesti. Possiede esclusivamente metodi statici.
- Classi Foundation: classi che si occupano della conversione dei risultati delle query sul DB in oggetti del Model e viceversa. Corrispondono 1:1 alle tabelle del database. Ad eccezione di FToken e FConfermaEmail, estendono FObject e possiedono tutte 3 attributi statici: table(nome della table), values(colonne della table), idString(stringa identificativa degli Id della tabella in questione). Un'ulteriore eccezione è rappresentata da FUtente, FCliente ed FAgenteImmobiliare: poiché le classi MCliente ed MAgenteImmobiliare sono sottoclassi di MUtente, e le relative table del database di conseguenza corrispondono fra loro, la totalità dei metodi è implementata da FUtente, ed FCliente e FAgenteImmobiliare possiedono esclusivamente i 3 attributi statici di cui sopra ed i relativi getters.
- FDataBase: singleton che si occupa dell'interazione con il DB tramite oggetti PDO. Contiene metodi generici, utilizzabili da tutte le classi Foundation mediante l'utilizzo degli attributi statici delle stesse. Esclusivamente nel caso di operazioni sulle table codice_conferma e token_login, si è ritenuto necessario utilizzare metodi non generici, in cui il nome della table e degli attributi è 'harcoded', in quanto i metodi per il codice di conferma sono contenuti nella classe FUtente.

PARTICOLARITÀ:

- Le password degli utenti, dell'Admin e i token vengono criptate al momento del salvataggio sul DB mediante l'algoritmo CRYPT_BLOWFISH di PHP, la verifica della password nel login e nel cambio Password avviene tramite password_verify() nelle rispettive classi Foundation.
- La classe FMedia funge da proxy, pur non essendolo formalmente, chiamando metodi della classe corrispondente all'ID dell'oggetto Mmedia passato

- MODEL:

Il 'core' di questo layer è dato dalla classe MAgenzia, contenente la lista dei clienti, degli agenti immobiliari, degli immobili e un oggetto di tipo MCalendario. Tramite i metodi checkDisponibilità di MAgenzia e addAppuntamento di MCalendario, è possibile verificare le fasce orarie disponibili per un dato cliente e un dato immobile in base al giorno e alla disponibilità degli agenti immobiliari e, dopo aver effettuato lo stesso controllo, aggiungere un appuntamento al calendario.
Il controllo del rispetto dei parametri(disponibilità dell'immobile, disponibilità dell'agente immobiliare, data e orario non passati) viene effettuato tramite gli MValidators, con un'implementazione del pattern Strategy.
PARTICOLARITÀ:
- La classe MData si occupa di fornire metodi facilmente accessibili per creare e modificare date, oltre a restituirle in formati diversi a seconda delle esigenze.

- CONTROLLER:

Il layer Controller è composto principalmente da classi corrispondenti alle URL chiamate dall'utente (CAdmin, CHome, CImmobile, CUtente). Esse si occupano di verificare se sono presenti i requisiti per attuare le richieste effettuate dagli utenti, di recuperare, modificare e aggiungere oggetti dal database e di comunicare alla view il risultato dell'esecuzione da mostrare all'utente. Oltre a quelle già citate, nel package controller sono presenti anche le classi CFrontController, CSessionManager e CMail.
La classe CMail ha la funzione di gestire l'invio di email preimpostate nel caso dell'invio all'utente del codice di verifica, di una nuova password o nel caso in cui gli si debba notificare una modifica ai suoi appuntamenti effettuata dall'admin. Per l'invio delle mail vengono utilizzati la classe PHPMailer e un indirizzo di posta gmail.

- VIEW:

Il layer View sfrutta, per la visualizzazione web, il template engine Smarty. Per agevolarne l'uso, è stata introdotta la classe VSmartyFactory che, in base al metodo richiesto, fornisce degli oggetti Smarty con variabili del template già assegnate.
Sono altresì stati introdotte due classi aventi il compito di "mascherare" ai controller le operazioni di invio e ricezione dati dalla View, agevolando così la manutenzione e fornendo un comodo strumento per futuri aggiornamenti.
La classe VReceiver fornisce metodi per il completamento di query incomplete e per ottenere dal server informazioni sul tipo di richiesta HTTP e sui parametri della stessa. Ove possibile restituisce ai controller oggetti del model.

Il VSenderAdapter è un'implementazione, poco canonica, del pattern Adapter. Ha la funzione di chiamare metodi delle classi View preposte alla costruzione di pagine web o di quelle preposte all'invio di package JSON a seconda del tipo di interfaccia utente. L'implementazione come singleton ha permesso inoltre di passare facilmente ai metodi delle classi View parametri aggiuntivi quali l'utente loggato e una stringa di errore. Non viene utilizzato per le funzioni dell'Admin.
La classe VImageReceiver si occupa invece di passare ai controller oggetti MMedia ottenuti tramite le immagini caricate dagli utenti.
Sono stati utilizzati, con numerose modifiche, due diversi template per la sezione utenti e per la sezione Admin(https://colorlib.com/wp/template/south/ – https://adminlte.io/). Per la visualizzazione del calendario è stato adottato https://fullcalendar.io/, creando un tema personalizzato. Script JS sono stati inoltre prodotti per la barra di ricerca degli immobili, per le funzioni di login e registrazione e per numerosi oggetti che producono particolari richieste HTTP POST.
Sono state inoltre utilizzate le API di Google Maps per la visualizzazione dei singoli immobili, tramite geocoding dell'indirizzo degli stessi.
La totalità delle pagine sono costruite in maniera modulare, includendo ai tpl della sezione utente header, footer e searchbar e 'wrappando' i tpl della sezione admin con header e sidebar.