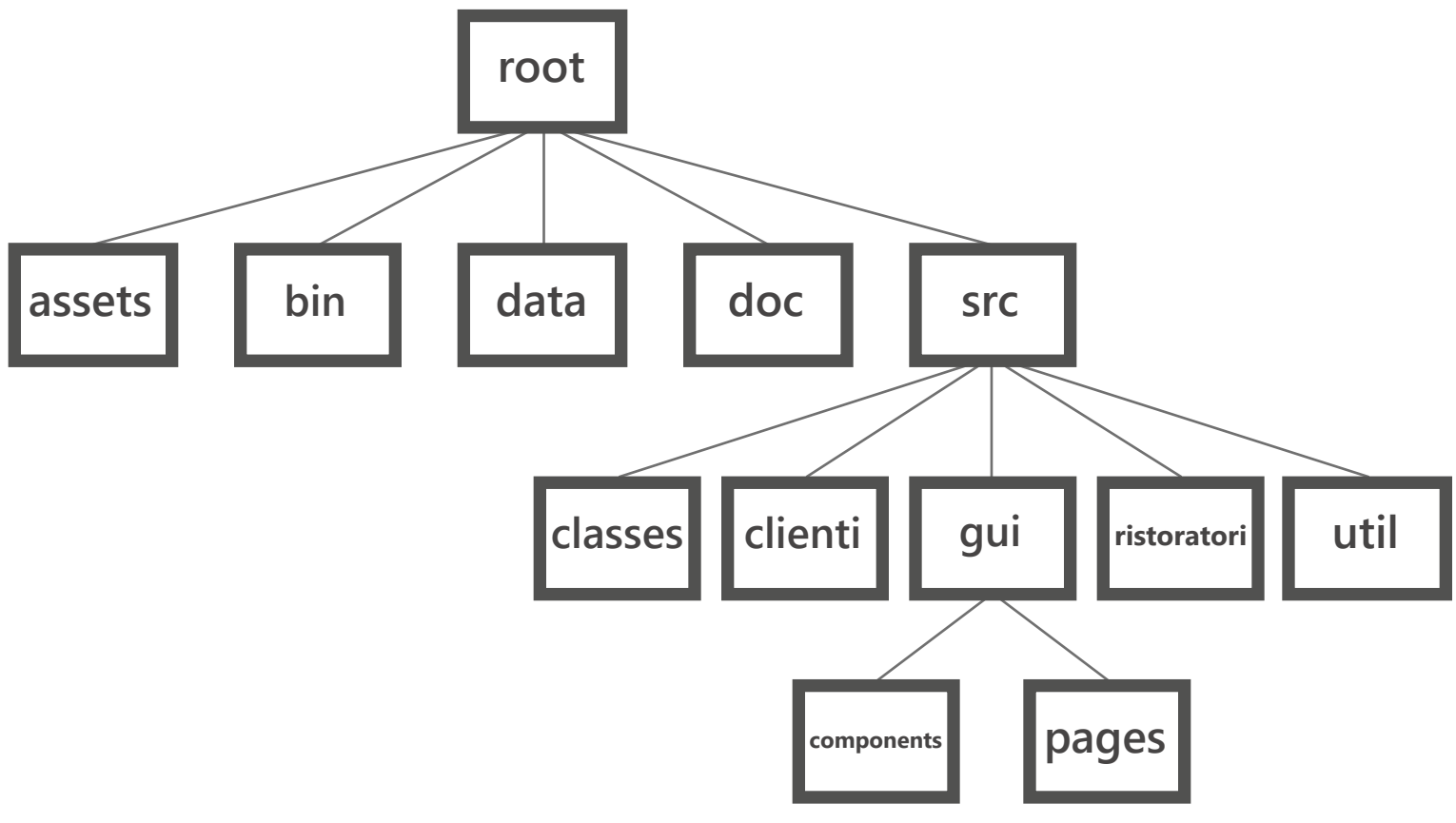
Loschiavo Christian 739894 Varese
Giubilei Ivan 739892 Varese
Rossi Nicolò 742626 Varese
Ferrario Andrea 740485 Varese

Manuale Tecnico

Descrizione dell'app FoodAdvisor

Struttura

La cartella principale del progetto contiene 5 sottocartelle: **assets, bin, data, doc, src**. La cartella **assets** contiene tutte le risorse utilizzare all'interno del programma (immagini, il font Manrope). La cartella **bin** contiene i file eseguibili ".class" del programma e i file eseguibili ".jar" per lanciare le app Clienti e Ristoratori in modo semplice per l'utente (con un doppio click). La cartella **data** contiene i due file di testo utilizzati per il salvataggio dei dati: il file EatAdvisor.dati contiene tutti le informazioni dei ristoranti e delle recensioni, mentre il file Utenti.dati contiene le informazioni degli utenti registrati. La cartella **doc** contiene i manuali utente e tecnico in formato pdf e il documento javadoc del codice. La cartella **src** contiene il codice sorgente del progetto.

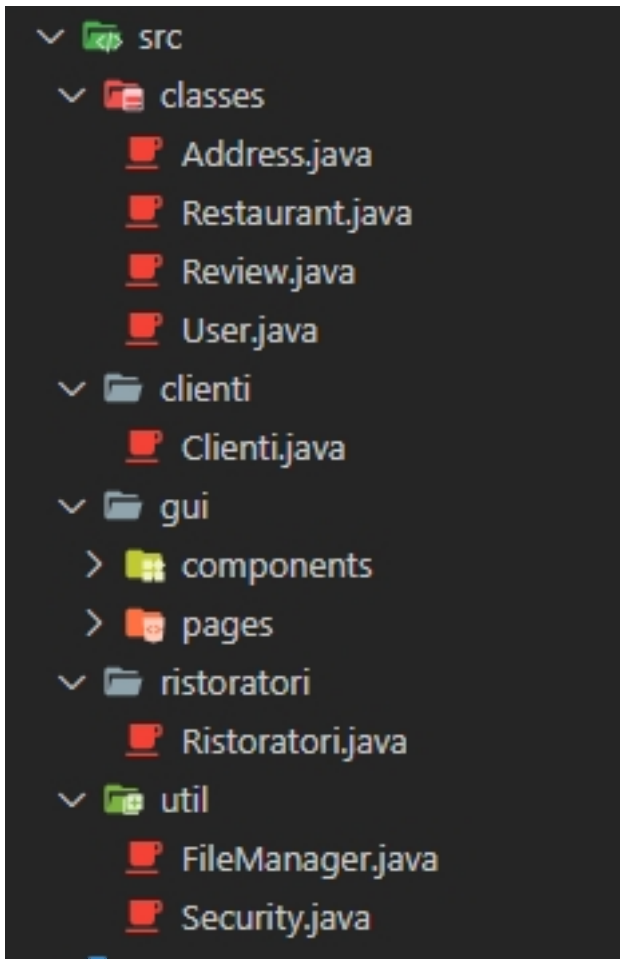


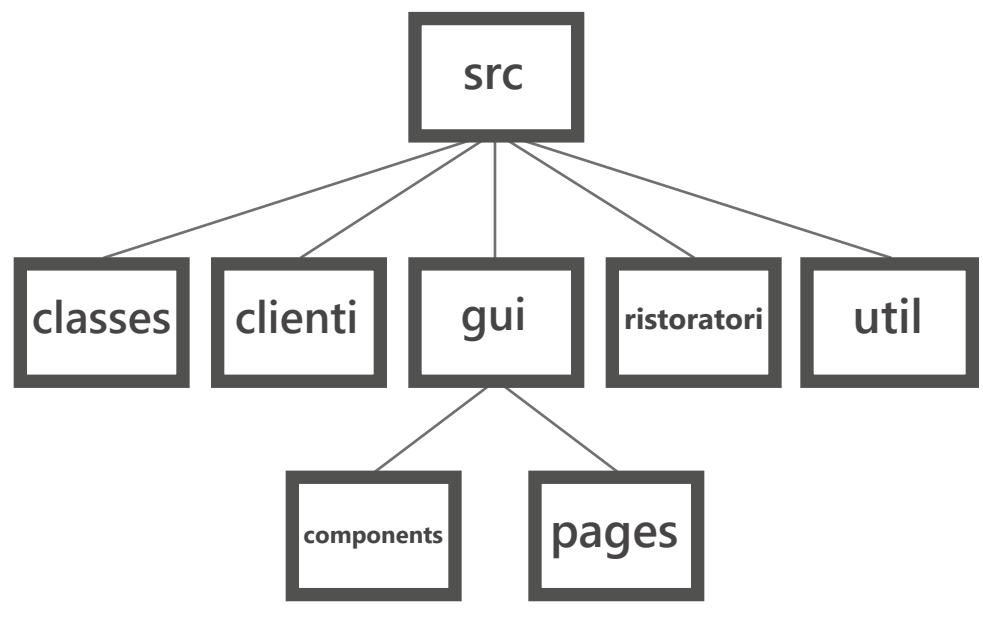
Struttura

Cartella src

La cartella **src** contiene altre 5 sottocartelle: **classes, clienti, gui, ristoratori, util**. Ognuna di queste cartelle è un **package**.

- Il package **classes** contiene 4 classi: **Address, Restaurant, Review, User**, contenute negli omonimi file .java. Queste classi sono state utilizzate come supporto in diverse parti del codice per evitare il passaggio di decine di parametri ai vari metodi.
- Il package **clienti** contiene il file Clienti.java, ovvero il file che contiene l'entry point all'applicazione dei clienti.
- Il package **gui** contiene due sottocartelle, denominate **"components"** e **"pages"** ed è utilizzata per contenere tutto ciò che riguarda l'interfaccia grafica.
- Il package **ristoratori** contiene il file Ristoratori.java, ovvero il file che contiene l'entry point all'applicazione dei ristoratori.
- Il package **util** contiene 2 classi statiche: **FileManager** e **Security,** contenute negli omonimi file .java e utilizzate rispettivamente per l'accesso ai file (sia dati che risorse) e per la cifratura delle password degli utenti.



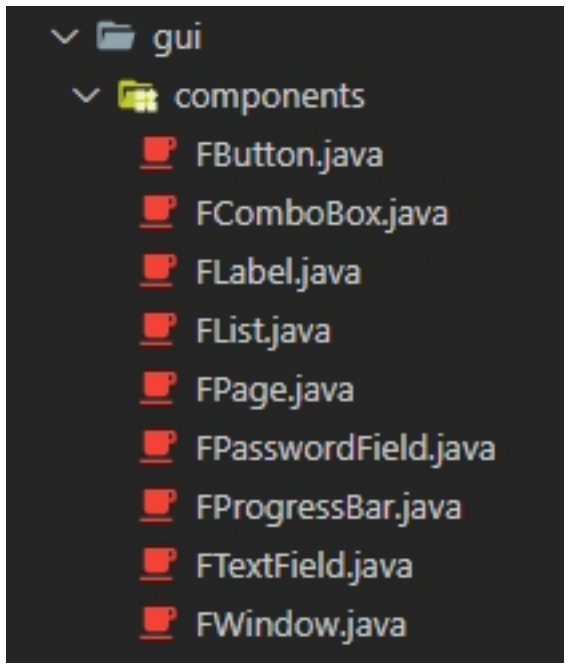


Struttura

Cartella gui

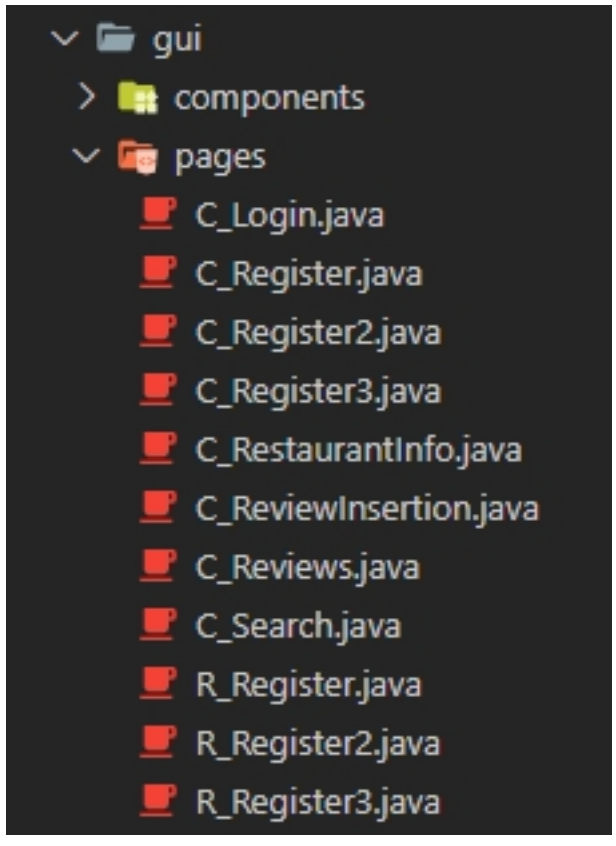
Il package **gui** è diviso in due sottocartelle: **components** e **pages**.

- components: Questa cartella contiene i componenti da noi modificati e personalizzati, partendo (estendendo) dai componenti base di swing.



FButton estende *JButton*
FComboBox estende *JComboBox*
FLabel estende *JLabel*
FList estende *JList*
FPage estende *JPanel*
FPasswordField estende *JPasswordField*
FProgressBar estende *JProgressBar*
FTextField estende *JTextField*
FWindow estende *JFrame*

- pages: Questa cartella contiene delle classi che possiedono come attributo un'istanza della classe "FPage", la logica utilizzata per cambiare pagine verrà spiegata nella prossima pagina.

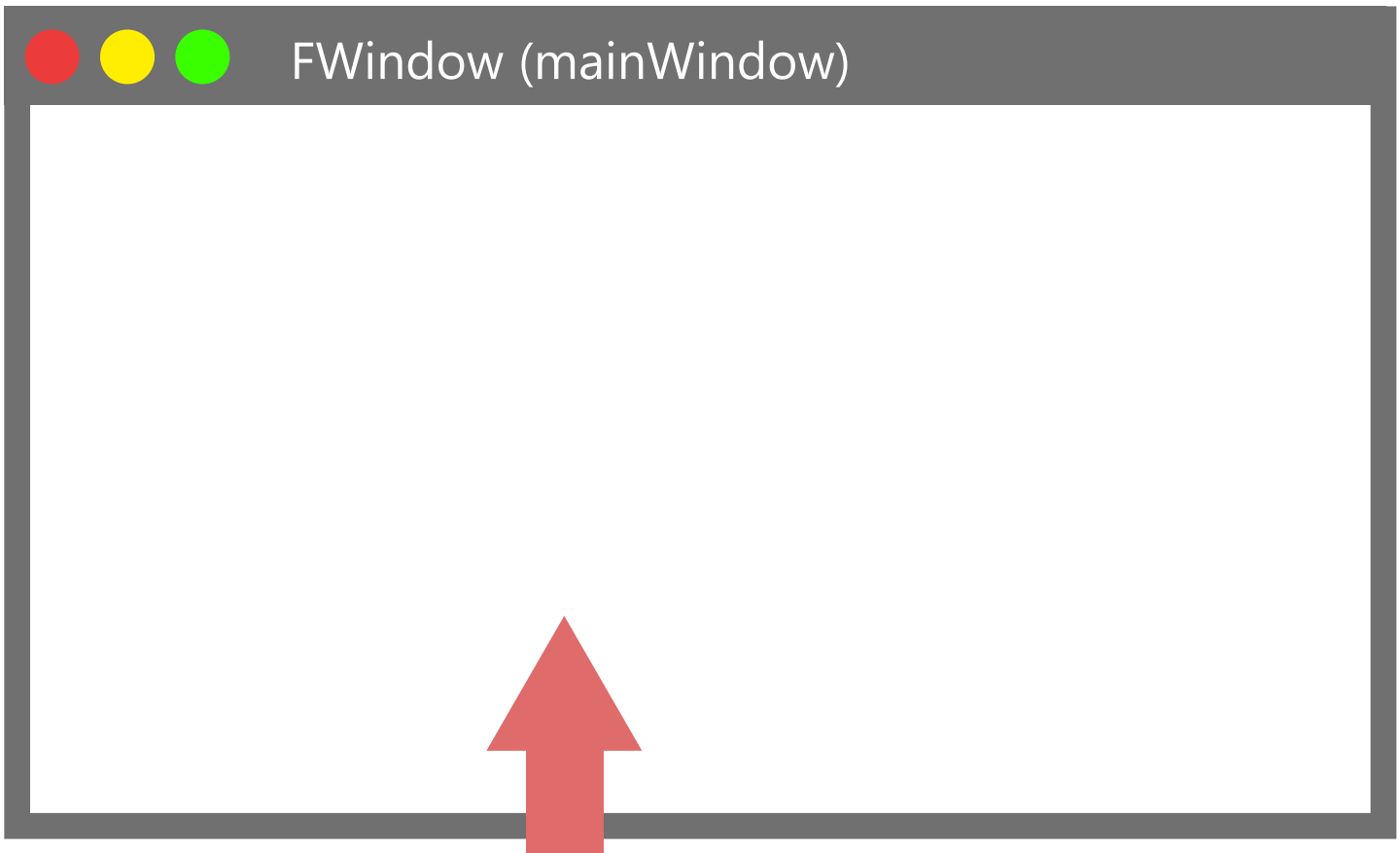


La nomenclatura utilizzata per le pagine prevede una lettera maiuscola iniziale che indica se la pagina sarà utilizzata nell'app dei clienti (C) oppure nell'app dei ristoratori (R), un underscore (_) e successivamente il nome della pagina, che deve indicare per cosa verrà utilizzata.

GUI

Utilizzo

L'interfaccia grafica è stata pensata in modo da avere una singola finestra (Main Window) all'interno del quale vengono tolte e inserite le pagine. Questa finestra è un'istanza della classe **FWindow** (il suo nome viene dall'utilizzo che ne facciamo) e al suo interno vengono inserite le istanze della classe **FPage**, presenti nella cartella **gui/pages.**



FPage (esempio)

Realizzata utilizzando i componenti presenti in **gui/components**

Dati

Cartella data

La cartella data contiene due file di testo: EatAdvisor.dati e Utenti.dati.
Il primo viene utilizzato per memorizzare le informazioni dei ristoranti e delle recensioni, il secondo viene utilizzato per memorizzare le informazioni degli utenti.

Abbiamo deciso di utilizzare dei file di testo perché abbiamo ritenuto più semplice ed efficiente processare le informazioni sia in scrittura che in lettura.

Record

Nel file Utenti.dati i record sono formati nel seguente modo:

nickname|nome|cognome|comune|provincia|email|password

il nickname agisce come chiave univoca del record.

Nel file EatAdvisor.dati sono contenute sia le informazioni relative ai ristoranti che le recensioni.
Il record del ristorante è formato nel seguente modo:

id_numerico|nome|tipo_via|nome_via|numero_civico|comune|provincia|CAP|numero_telefono|sito_web|tipologia_ristorante

Il record di una recensione è formato nel seguente modo:

id_ristorante|nickname|valutazione|titolo|descrizione

Utilità

Cartella util

Il package **util** contiene due classi statiche: **FileManager** e **Security**.

Nella classe **FileManager** abbiamo diversi metodi che vengono utilizzati per l'accesso, sia in lettura che in scrittura, ai file dati e alle risorse.

Nella classe **Security** abbiamo implementato il protocollo di cifratura **SHA1** per cifrare le password degli utenti prima di memorizzarle e garantire così una maggiore sicurezza.

```java
/**
 * Cifra la stringa inserita utilizzando l'algoritmo SHA1
 * @param plaintext Stringa da cifrare
 * @return Stringa cifrata con sha1 */
public static String GetHash(String plaintext) {
    // Per poter utilizzare gli algoritmi di hashing è
    // necessario lavorare con i bytes, quindi preleviamo
    // i bytes della stringa usando il metodo getBytes()
    byte[] inputBytes = plaintext.getBytes();
    // Inizializzazione del valore di ritorno
    String hashValue = "";
    // Utilizzo del StringBuilder per lavorare in modo più
    // efficiente sulle stringhe
    StringBuilder sb;

    try {
        // Chiamo il costruttore di StringBuilder
        sb = new StringBuilder();
        // Seleziono l'algoritmo di cifratura desiderato
        MessageDigest msgDigest = MessageDigest.getInstance("sha1");
        // Applica l'algoritmo al byte buffer
        msgDigest.update(inputBytes);

        // Completa l'algoritmo
        byte[] digestedBytes = msgDigest.digest();
        // Convertiamo i bytes in hex string
        for(byte b: digestedBytes)
            hashValue = sb.append(String.format("%x", b)).toString();

    } catch(Exception e) {
        System.out.println("Errore Security :38");
        System.exit(0);
    }

    // Ritorno la stringa in SHA1
    return hashValue;
}
```

L'algoritmo di hashing **SHA1** permette di ottenere una stringa di dimensione fissa partendo da un altro dato, non necessariamente una stringa, perché lavora a livello di bit.
È una funzione unidirezionale (irreversibile) vale a dire che una volta generato il testo cifrato, non sarà possibile ottenere quello originale in nessun modo.
Passando come input lo stesso dato avremo sempre lo stesso valore in output, grazie a questa proprietà siamo in grado di confrontare l'hash della password memorizzata con quella che viene inserita al momento del login.

Enrty Points

Classi principali

Per mantenere il costruttore della classe Clienti pulito e più facilmente gestibile abbiamo deciso di creare metodi appositi per aggiungere i listener necessari ai componenti delle pagine man mano che queste venivano istanziate.
Nel costruttore rimane per cui la registrazione del font utilizzato nelle app, l'ottenimento dei ristoranti contenuti nel file EatAdvisor.dati, il setup della mainWindow e l'inizializzazione e inserimento della pagina di login nella mainWindow con l'aggiunta dei listener necessari.

```
Run | Debug
public static void main(final String[] args) {
  SwingUtilities.invokeLater(() -> new Clienti());
}

/** Costruttore della classe clienti */
public Clienti() {
  // setup della window
  registerFonts();
  user = null;
  restaurants = FileManager.GetRestaurants();

  mainWindow = new FWindow("FoodAdvisor Clienti");
  loginPage = new C_Login();

  changePage(loginPage.getPage());
  addLoginPageListeners();

  mainWindow.setVisible(true);
  loginPage.getPage().requestFocusInWindow();
}
```

Per ragioni di efficienza abbiamo deciso di mantenere come proprietà della classe Clienti (Ristoratori) delle FPage non inizializzate che vengono successivamente istanziate il momento prima di essere caricate nella mainWindow.

In questo modo, ad esempio, inizializziamo la pagina di login e la inseriamo nella mainWindow, se clicchiamo sul pulsante "Registrati" in questa pagina, verrà istanziata la prima FPage di registrazione e verranno aggiunti, tramite l'apposito metodo, i listener necessari in quella pagina.