Copia locale del registro delle attività

Data	Ore spese	Totale ore	Descrizione	Lunghezza	
13/12/2019	1,20	1,20	L'applicazione è un gestionale di spese personali/familiari: l'utente inserisce le spese che effettua durante un certo periodo di tempo (configurabile) e la categoria di spesa a cui appartiene la spesa e il sistema fornisce una visualizzazione grafica delle spese. Realizzazione del mockup, composto da: una tabella per visualizzare le spese, un form per inserire nuove spese, un grafico a torta con le categorie di spesa, le spese totali. Definizione del caso d'uso di "Inserimento di una spesa"	496	-4
14/12/2019	1,04	2,24	Aggiunta del titolo e del pulsante di ELIMINA al mockup. Aggiunto caso d'uso per eliminare una spesa inserita. Definizione degli elementi del file di configurazione, in particolare: colori, ip e porta dei server di backlog e database, periodo temporale, categorie di spese. Definizione di cache locale: memorizza i dati inseriti nel form, ma non ancora inviati. Archiviazione: spese inserite dall'utente dal form. File di log remoto: avvio e terminazione dell'applicazione, pressione dei pulsanti	496	-4
14/12/2019	0,95	3,19	Aggiunta la possibilità per l'utente di selezionare il periodo temporale di cui bisogna mostrare i dati. Modificato il mockup per consentirne la scelta attraverso due Date Picker e un pulsante. Rimossa l'opzione di configurazione che prevedeva di settare tale parametro, poiché è mutevole nel tempo. Aggiunto il caso d'uso per scegliere il periodo temporale. Modificato il file di log per includervi la selezione del pulsante "CAMBIA IL PERIODO".	446	-54
16/12/2019	0,74	3,93	Aggiunta la sezione di "login", in cui l'utente può inserire il proprio nome utente per vedere i propri dati (senza autenticazione via password). Modificato il mockup con una casella di input per il nome utente e un pulsante. Aggiunto caso d'uso per il corretto login. Modifica della cache locale e dell'archiviazione, per includervi il nome utente. Aggiunto evento di pressione del pulsante "LOGIN" al file di log. Aggiunta opzione per il colore del pulsante "LOGIN" al file di configurazione.	494	-6
17/12/2019	0,49	4,42	Modifiche del progetto come da accordi presi al ricevimento. Modificato pulsante LOGIN in CARICA DATI. Aggiunto drop-down menu per selezionare la Categoria. Aggiunta opzione di configurazione che permette di impostare il periodo temporale di default. Modifiche minori alla documentazione, in conseguenza al cambio di nome del pulsante LOGIN. Aggiunto evento scatenante del log: selezione di una riga della tabella.	414	-86
12/03/2020	1,02	5,44	Individuazione delle classi principali e divisione delle stesse sui tre livelli di front-end, middleware, back-end, con prima descrizione sommaria. IntefacciaGestoreSpese: interfaccia grafica che contiene tutti gli oggetti grafici aggiuntivi (TabellaSpese, GraficoCategorieDiSpesa...). GestoreSpese: controller applicativo; inizializza GUI e gestisce eventi. Classi standard come i gestori di cache, file e database, classe Cache serializzabile, classi incaricate alla serializzazione XML e al log	497	-3
14/03/2020	1,69	7,13	Aggiunta di descrizioni più esaustive alle classi sul documento di progetto. Aggiunta di una classe VoceSpesaBean, utilizzata da TabellaSpese per aggiornare la TableView. Inizio del diagramma delle classi con il software StarUML. Design completo delle classi GestoreSpese (estende Application), Cache (implementa Serializable; tutti attributi pubblici e costruttore), GestoreCache (metodi per carica e salvare la cache), GestoreFile (metodi per caricare e salvare su file testuali) e VoceSpesa.	494	-6
15/03/2020	1,98	9,11	Design UML di: ParametriDiConfigurazione (legge il file di configurazine XML e ne mette a disposizione i parametri), GestoreDatabase (metodi per leggere, inserire ed eliminare voci di spesa), LogEventoXML (offre metodi per la costruzione e la serializzazione XML di un evento di log), GestoreLogEventiGUI (invia al server un evento di log), ServerLogEventiGUI (utilizza i metodi di GestoreFile per memorizzare i log ricevuti), ValidatoreXML (valida un file xml secondo uno schema specificato).	493	-7
18/03/2020	0,71	9,82	Modifica nel modello UML: sostituzione di tutti gli attributi di tipo Date con tipo LocalDate. Accorpamento in LogEventoXML di data e ora in un unico attributo LocalDateTime. Design della classe VoceSpesaBean, che conterrà i dati di una riga della TabellaSpese. Introduzione di alcune dipendenze tra classi middleware e backend. Inizio ricerca e prototipazione riguardo alle TableView editabili, così da poter definire il design di FormDiInserimento.	450	-50
27/03/2020	4,33	14,15	Redesign di GestoreSpese aggiungendo metodi middleware per passaggio informazioni tra frontend e backend, in coerenza con la sua responsabilità di controllore applicativo. Design di 5 classi frontend, come InterfacciaGestoreSpese (che instanzia le altre classi grafiche e gestisce gli eventi di modifica delle voci) e GraficoCategoriaDiSpesa, con metodi per modificare/aggiornare i dati visualizzati. Protipazione delle classi PieChart e Pair per il loro corretto utilizzo in GraficoCategoriaDiSpesa	499	-1
28/03/2020	0,59	14,74	Modifica dei costruttori di molte classi frontend, così da aggiungere un riferimento alla classe frontend principale "InterfacciaGestoreSpese". Aggiunta delle dipendenze <<has>> con la suddetta classe. Aggiunta delle dipendenze <<use>> con la classe GestoreSpese per le classi frontend che utilizzano i metodi middleware offerti per aggiungere/eliminare voci dal sistema e aggiornare i dati visualizzati sull'interfaccia. Modifiche minori allo schema UML per aumentarne la leggibilità.	485	-15
29/03/2020	0,63	15,37	Prototipazione di javafx.scene.chart.PieChart, per verificare se l'utente può configurare i colori delle categorie del grafico. Il test ha avuto successo. Aggiunta voce nel file di configurazione XML, relativa al numero massimo di categorie da mostrare nel grafico, privilegiando quelle con percentuale più alta, per aumentare la leggibilità (le restanti verranno accorpate in categoria "Altre"). I colori configurabili sono relativi alle categorie mostrate. Modifica conseguente dello schema UML.	497	-3
01/04/2020	1,63	17,00	Rifinitura delle descrizioni delle classi, in particolare per InterfacciaGestoreSpese e GestoreSpese per le quali, essendo classi molti grandi, non erano ancora chiari i limiti delle responsabilità. Schema UML: rese statiche le classi GestoreDatabase e GestoreLogEventoGUI; aggiunta di dipendenze tra le due classi con ParametriDiConfigurazione; aggiunta di alcune dipendenze minori; miglioramenti estetici per la leggibilità; divisione in tre colori delle classi UML (frontend, middleware, backend)	499	-1
02/04/2020	2,29	19,29	Design del database, costituito dalle tabelle utente e spesa, con trigger per garantirne la consistenza; la chiave di spesa è il nome utente e un indice numerico (l'n-esima spesa dell'utente X avrà indice n). Traduzione in Java di tutte le classi del modello (attributi e metodi senza implementazione). Implementazione del metodo ottieniVociSpesa di GestoreDatabase: interroga il database per ottenere le spese di un periodo temporale ben definito e le restituisce come una lista di oggetti VoceSpesa	500	0
03/04/2020	3,69	22,98	Implementazione di GestoreFile (carica e salva file testo dalla cartella files), GestoreDatabase (estrae dal database i dati da mostrare sulla GUI), GestoreLogEventiGUI (invia i log al server), LogEventoXML e ValidatoreXML. Conversione dei metodi di GestoreDatabase e GestoreLogEventiGUI a NON statici e aggiunta di attributi di configurazione (settati dal costruttore). Queste classi verranno instanziate da GestoreSpese, inizializzandole con i parametri di configurazione, come suoi attributi.	494	-6
05/04/2020	2,39	25,37	Aggiunti i commenti alla classe ValidatoreXML e rifinitura commenti già inseriti. Implementazione totale di VoceSpesa, Cache (viene serializzato su file) e GestoreCache (preleva i dati che necessita l'oggetto Cache dagli elementi della GUI in caso di salvataggio e si occupa di inserirli negli stessi elementi in caso di caricamento). Implementazione parziale del metodo start di GestoreSpese, in cui vengono costruiti la GUI e i vari gestori, nonché estratti i parametri di configurazione e la cache	500	0

06/04/2020	2,18	27,55	Implementazione di GestoreSpese: start (aggiunta gestione eventi); inserisciNuoveSpese, eliminaVoceSpesa, inviaLogEvento (propagano la richiesta ai gestori, dopo aver recuperato i dati mancanti); aggiornaDatiSpeseGUI (elabora i dati del database e li passa agli elementi grafici responsabili della loro visualizzazione). Aggiunto metodo a GestoreDatabase per ottenere la spesa totale per ogni categoria. Modifica al metodo backend che invia i log: recupera autonomamente l'indirizzo IP del client.	497	-3
07/04/2020	2,78	30,33	Implementazione di InterfacciaGestoreSpese: la classe si occupa di istanziare gli elementi grafici della GUI e ne configura l'aspetto grafico e la disposizione spaziale; crea gli handler per gli eventi di interazione con i pulsanti INSERISCI, ELIMINA, ANNULLA. Implementazione di BarraCaricaDati: permette all'utente di inserire il proprio nome utente per caricare i suoi dati; gestisce l'hander per tale evento, il quale lancia la routine per caricare i dati e invia un log dell'evento.	487	-13
10/04/2020	1,86	32,19	Sviluppo di VoceSpesaBean. TabellaSpese: costruzione della TableView e configurazione aspetto; implementato il metodo che ricarica la tabella e quello che elimina una voce selezionata; eliminato il metodo che inserisce una voce spesa, poiché inutilizzato. Inizio sviluppo di FormDiInserimento: decisione di abbandonare l'implementazione con la TableView poiché richiederebbe lo sviluppo di CellFactory per le celle che selezionano la categoria e la data; utilizzo dei UI controls offerti da JavaFX.	498	-2
10/04/2020	2,44	34,63	FormDiInserimento: la classe utilizza alcuni UI controls di JavaFX (TextField, ChoiceBox, DatePicker) per permettere all'utente l'inserimento dei dati di una voce spesa; le opzioni del ChoiceBox "Categoria" sono configurabili dall'utente; metodi per flush, estrarre i dati inseriti dall'utente e mostrare i dati memorizzati in cache all'ultima chiusura. GraficoCategorieDiSpesa: viene utilizzata la classe PieChart per il grafico a torta (i colori sono configurabili dall'utente).	481	-19
12/04/2020	1,53	36,16	Sviluppo di ZonaCambioPeriodo, che permette all'utente di cambiare il periodo di analisi dalla GUI. Modifiche minori alla disposizione spaziale degli elementi della GUI. Modificata la funzionalità che preleva le coppie Categoria-Importo per il grafico, cosicché il database ne restituisca il numero configurato dall'utente. Risolto un errore in GestoreDatabase dovuto all'omonimia di java.sql.Date e java.util.Date, che impediva la conversione corretta in LocalDate. Inizio sviluppo del server log.	498	-2
18/05/2020	2,89	39,05	Schema XSD di Parametri di Configurazione e LogEventoXML: particolare attenzione per la corretta validazione dei colori JavaFX, degli indirizzi IPv4 e dei timestamp. Sviluppo di ParametriDiConfigurazione, che mantiene in memoria i parametri scritti dall'utente su file XML. Scelta di progettazione per ParametriDiConfigurazione: i parametri nel file XML sono tutti opzionali da inserire per l'utente; quelli mancanti verranno inizializzati al valore di default	460	-40
29/05/2020	4,86	43,91	Sviluppo meccanismo di personalizzazione dei colori del grafico: GestoreSpese estrae i parametri e li scrive in un formato appropriato su un file css, collegato all'applicazione. Test dell'applicazione e correzione degli errori (validazione XML errata, creazione log non corretti, aggiunta nuovo utente difettosa...) . Inizio nuovo ciclo. UML: Aggiunta dei package, segmentazione metodi pubblici troppo lunghi, eliminazione dipendenze non utilizzate. Modifica conseguente del codice (segmentazione).	500	0
01/07/2020	1,08	44,99	Collaudo finale per verificare che non vi siano stati errori nella modifica del codice. Scrittura del manuale utente, in cui vengono eseguiti ed illustrati nella documentazione i casi d'uso definiti in fase di analisi	218	-282