UNIONE EUROPEA

Ministero dello Sviluppo Economico

Regione Puglia
Dipartimento Sviluppo Economico, Innovazione, Istruzione, Formazione e Lavoro

PUGLIA FESR·FSE 2014/2020
Il futuro alla portata di tutti

BANDO INNOLABS

"Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale"

Y95B457 Progetto

serviziLocali S.p.A.

SBSoft

UNIVERSITÀ DEL SALENTO

easyPAL

Deliverable D.D1 – Report: Piano della sperimentazione

Data 04/10/2019

Versione V2.0

NOME DEL DOCUMENTO

Deliverable D.D1 – Report: Piano della sperimentazione

INFORMAZIONI GENERALI

Nome progetto	EasyPAL “Ecosistemi e Servizi Digitali in Cloud per i Cittadini e la PA Locale”
Ambito	BANDO INNOLABS “Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale”
Riservatezza	Riservatezza ai sensi dell'art. 14 dell’atto di costituzione dell’ATS denominata “Easy PAL” registrata a Lecce in data 01/06/2018 al n. 5694/1T da Notaio Pellegrino.

RESPONSABILITA'

Funzione	Nome	Data
Redatto da	Unisalento	16/09/2019
Contributi di	Unisalento, Servizi Locali, SbSoft	
Controllato e approvato da	Unisalento	04/10/2019

INDICE

Indice delle tabelle

1. Premessa

Il deliverable presenta la pianificazione della fase di sperimentazione e test in vivo delle piattaforma e dei servizi da essa esposti. La fase di test consta di due parti, la prima finalizzata alla verifica dei vari moduli che costituiscono piattaforma EasyPAL, la seconda parte relativa alla verifica delle funzionalità dei prototipi sviluppati, con particolare riferimento alla User Interface. Le due fasi saranno denominate nel seguito "Testing della piattaforma EasyPAL" e "Testing dei servizi applicativi prototipali", rispettivamente.

Entrambe le fasi di test saranno condotte mediante metodologie già affermate nei contesti scientifici.

La piattaforma EasyPAL sarà testata mediante le seguenti categorie di test: Testing Strutturale, Unit Testing, Penetration Test.

Il *Testing Strutturale*, o *White Box*, verte alla verifica del corretto funzionamento dei macro componendi del sistema (Microservizi erogati in cloud, Motore Ente Digitale, Motore di Workflow e Documentale, Interfacce Mobili e Interfacce Web).

Lo Unit Testing, o Black Box Testing, è una modalità molto diffusa di testing a granularità fine, il cui scopo è verificare i dettagli relativi alle singole unità di codice, come ad esempio le Classi.

La fase di *Penetration Test consiste nella valutazione del livello di* sicurezza del sistema mediante la simulazione di attacchi da parte di un utente malintenzionato. Si rende necessaria data la natura del sistema trattato, trattandosi di un sistema web, peraltro organizzato a Microservizi che comunicano con interfacce Rest: garantire la sicurezza dagli accessi non autorizzati è fondamentale. Come anticipato, l'analisi è condotta dal punto di vista di un potenziale attaccante e consiste nello sfruttamento delle vulnerabilità rilevate al fine di ottenere più informazioni possibili per accedere indebitamente al sistema.

Il test dei servizi applicativi sarà effettuato adottando il framework "MILE", un metodo di misurazione delle qualità di applicazioni interattive che usa tecniche euristiche e di osservazione degli utenti.

Durante la fase di test si prevede, complessivamente, il coinvolgimento di almeno 110 utenti finali, in particolare:

- Cittadini (almeno n. 75);
- Operatori comunali, a loro volta suddivisi in:
    - Responsabili IT e dirigenti degli Enti (almeno n. 5);
    - Funzionari amministrativi degli Enti (almeno n. 10);
    - Rappresentanti delle associazioni di categoria (almeno n. 20);

Il presente deliverable descrive nel dettaglio modalità, tempistiche e strumenti con i quali verrà condotta la sperimentazione.

2. Caratterizzazione del testing della piattaforma EasyPAL

L'insieme degli strumenti per il testing dei software è fondamentale non solo per esaminare il funzionamento delle varie parti del sistema al momento della loro scrittura, ma anche per disporre di uno strumento formale che eviti un problema fondamentale: i processi di aggiornamento del codice non devono produrre regressioni, cioè malfunzionamenti in parti del sistema correttamente funzionanti prima dell'incorporazione degli stessi aggiornamenti. Da qui nasce l'esigenza disporre di strumenti automatici di test, da lanciare a seguito di ogni insieme di modifiche e preliminarmente alle build di sistema. Pertanto, come logico attendersi, la presenza di questi test riduce i fattori di rischio intrinseci a ogni processo di modifica e aumenta il livello di sicurezza del team di sviluppo.

Come già esplicitato in premessa, i test implementati per la validazione della piattaforma sono categorizzati come Unit Testing, Testing Strutturale e Penetration Test, le quali saranno descritte nelle successive sezioni del presente capitolo. Saranno inoltre discussi gli strumenti software selezionati per l'implementazione e l'esecuzione degli stessi test.

Per una trattazione della pianificazione temporale dell'esecuzione dei test della piattaforma, si rimanda alla Sezione 4 del Deliverable.

2.1 Unit Testing

Con il termine test di unità ci si riferisce alle procedure utilizzate per verificare che un particolare frammento di codice presenti il comportamento aspettato. L'obiettivo dei test di unità è quindi di verificare che ogni classe presenti il comportamento atteso, indipendentemente dal contesto di utilizzo[1].

La completa esecuzione senza errori dei test di unità rappresenta l'evento di avvio dei test di integrazione che, come si vedrà nel paragrafo successivo, servono a verificare che componenti funzionanti singolarmente, una volta correttamente assemblati tra loro diano luogo a elementi più grandi ancora funzionanti correttamente.

Un'opportuna implementazione dei test di unità offre un'indubbia serie di vantaggi.

Individuare rapidamente gli errori. In particolare, questi test permettono di individuare eventuali malfunzionamenti, non appena redatto il codice: frequentemente, errori individuati in fasi successive del processo di sviluppo del software, durante i test di accettazione o addirittura a sistema in produzione, risultano difficili da investigare, da risolvere e, in ultima analisi, decisamente più onerosi.

Aumentare la qualità del codice. La verifica estensiva del codice permette di individuare e risolvere un'elevata quantità di errori, spesso anche quelli più nascosti, e quindi di garantire una maggiore qualità del sistema nel suo complesso. Ciò, inoltre, semplifica la produzione del codice basato su quanto già prodotto, aumentando, contestualmente, il grado di confidenza degli stessi programmatori.

Semplificare il processo di refactoring. La presenza di estesi test di unità permette di minimizzare gli inevitabili fattori di rischio connessi a estesi processi di refactoring e, allo stesso tempo, aumenta il livello di sicurezza del team di sviluppo.

Semplificare il processo di integrazione. Sebbene la pertinenza dei test di unità, per loro stesso principio, non riguardi l'integrazione di parti del sistema, il fatto che ogni elemento sia stato verificato in dettaglio, prima di essere assemblato in elementi più grandi, riduce il livello di incertezza e quindi semplifica e velocizza il processo stesso di integrazione.

Migliorare la documentazione. In alcuni casi, lo studio di determinate API o di parti di codice è notevolmente semplificato dall'analisi di esempi concreti come quelli implementati nei test di unità. Pertanto, tali test possono essere considerati a tutti gli effetti come parte integrante della documentazione del codice, oltretutto assolutamente in linea con l'ultima versione del codice.

2.2 Testing Strutturale

Il Testing White Box è un testing strutturale, poichè utilizza la struttura interna del software per ricavare i dati di test.

Tramite il testing White Box si possono formulare criteri di copertura più precisi di quelli formulabili con testing Black Box, fornendo inoltre maggiori indicazioni di debugging.

Nella strategia testing strutturale viene presa in considerazione la struttura interna del sistema, analizzando il sistema in modalità, appunto, white-box.

La costruzione dei test-case deriva da un esame formale della logica interna del sistema, composta da algoritmi e strutture dati, senza analizzare il sistema nel suo complesso. Può essere interessante capire come sia possibile soddisfare il criterio di ottimalità, ovvero come sia possibile ricavare un insieme di test in grado di garantire il ritrovamento di tutti gli errori presenti nel sistema.

Una prima ridefinizione del criterio di ottimalità potrebbe essere la richiesta che ogni istruzione del programma venga eseguita almeno una volta, ma la sola esistenza di un ciclo all'interno del codice può generare errori che questo criterio non evidenzia.

Il criterio ottimale viene allora definito come la richiesta che tutti i cammini all'interno del grafo di controllo del programma vengano percorsi, ovvero la ricerca esaustiva di tutti i cammini distinti esistenti nel programma. Se quindi la ricerca esaustiva non è proponibile, risulta necessario ricercare un insieme ottimale di cammini che assicurino la maggior probabilità di rilevare gli errori presenti nel software, con il minimo dei test. La ricerca dei cammini, sia essa esaustiva o meno, non assicura comunque che il software sia conforme alle specifiche del sistema, così come non è in grado di rilevare errori legati a particolari valori dei dati.

Analizzando a grandi linee i criteri che rientrano all'interno della strategia White-Box, il criterio di copertura delle istruzioni o statement coverage si prefigge lo scopo di trovare un insieme di test tale che sia garantito che ogni istruzione all'interno del software sia eseguita almeno una volta.

Copertura delle decisioni. Il criterio di copertura delle decisioni o decision coverage, nota anche con il nome di branch coverage, parte dal presupposto che è necessario esaminare ogni ramificazione all'interno del grafo sia per il valore di verità che per il valore di falsità della ramificazione.

Copertura delle condizioni. Non sempre però la copertura delle decisioni porta a test in grado di rilevare tutti gli errori, per cui si ricorre alla copertura delle condizioni o condition coverage.

Per soddisfare questo criterio, si deve scrivere un insieme di test-case tali che ciascuna condizione elementare all'interno di una decisione assuma tutti i possibili valori almeno una volta.

Come per la copertura delle decisioni, si dovrà affiancare questo criterio con la copertura delle istruzioni, al fine di verificare anche le possibili eccezioni nell'esecuzione dei test.

Copertura delle decisioni e condizioni. La copertura delle condizioni non implica però la copertura delle decisioni, quindi si dovrà fare ricorso alla copertura delle condizioni e decisioni.

Per soddisfare il criterio si deve ricercare un insieme di test-case tali da garantire che ogni condizione assuma il valore di verità e falsità almeno una volta e che ogni decisione all'interno del software sia percorsa almeno una volta sia per il ramo vero che per il ramo falso.

Questo potrebbe apparire un criterio adeguato per il test strutturale di un software, ma bisogna tenere conto che di norma una macchina non è in grado di verificare condizioni multiple senza spezzarle in più condizioni singole.

Copertura delle condizioni multiple. Concludendo, laddove sia necessario che l'insieme di test-case preveda tutti i possibili valori di ciascuna combinazione di condizioni presenti in una decisione, interviene il criterio di copertura delle condizioni multiple.

Anche in questo caso si dovrà affiancare il criterio di copertura delle istruzioni, al fine di garantire anche la copertura nei casi eccezionali prima elencati.

2.3 Penetration Test

Il Penetration Test si compone di una serie di attacchi informatici simulati al sistema informatico per verificare la presenza di vulnerabilità sfruttabili. Nel contesto della sicurezza di applicazioni front-end, siano esse di tipo web o mobile, basate su servizi di backend, il test può comportare il tentativo di violazione di un numero qualsiasi di sistemi/moduli applicativi per scoprirne la vulnerabilità [4].

Le informazioni fornite dal test di penetrazione possono essere utilizzate per ottimizzare le politiche di sicurezza di un Web Application Firewall (WAF) e correggere le vulnerabilità rilevate.

2.3.1 Fasi

Il Penetration Test può essere suddiviso in cinque fasi:

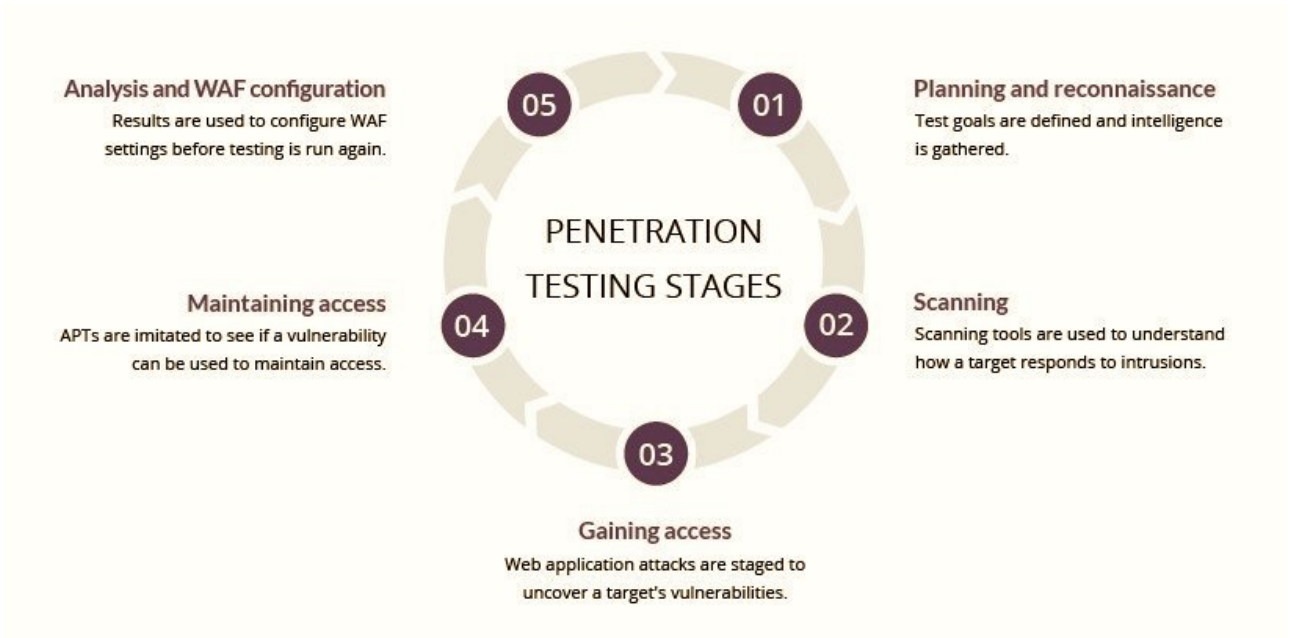

Figura 1 - Fasi del Penetration Testing

1. Pianificazione e ricognizione. La prima fase prevede:

- Definizione dell'ambito e degli obiettivi di un test, compresi i sistemi da affrontare e i metodi di test da utilizzare;
- Raccolta di informazioni architetturali (ad es. Nomi di rete e di dominio, server di posta) per comprenderne meglio il funzionamento e le sue potenziali vulnerabilità.

2. Scansione. Nel secondo step si tenta di capire come l'applicazione di destinazione risponderà ai vari tentativi d'intrusione. Questo in genere viene fatto analizzando il codice dell'applicazione in maniera dinamica e/o statica, a seconda che l'ispezione venga eseguita durante l'esecuzione o al di fuori di essa, rispettivamente.

3. Ottenere l'accesso. Rappresenta la fase cruciale del test poiché in essa vengono lanciati gli attacchi veri e propri al sistema. I tester cercano quindi di sfruttare queste vulnerabilità, in genere aumentando i privilegi, rubando dati, intercettando il traffico, ecc., per determinare l'entità dei danni causabili.

4. Mantenimento dell'accesso. L'obiettivo di questa fase è stabilire se la vulnerabilità può essere utilizzata per ottenere una presenza persistente nel sistema sfruttato. L'idea è di imitare le minacce persistenti avanzate, che spesso rimangono in un sistema per lungo tempo al fine di sottrarre i dati più sensibili di un'organizzazione.

5. Analisi. I risultati del Penetration Test vengono quindi compilati in un rapporto dettagliato:

- Vulnerabilità specifiche che sono state sfruttate
- Dati sensibili a cui si è potuti accedere

- Il periodo di tempo durante il quale il tester ha mantenuto una permanenza persistente nel sistema, senza essere rilevato.

2.3.2 Classificazione dei Penetration test

Di seguito vengono riportate le categorie di Penetration test comunemente utilizzate.

Test esterni. I test di penetrazione esterni hanno come obiettivo le risorse di un'azienda visibili in rete, cioè quelle presenti nell'area DMZ (DeMilitarized Zone), ad esempio l'applicazione Web stessa, il sito Web dell'azienda e i server di posta elettronica e nomi di dominio (DNS). L'obiettivo è ottenere l'accesso ed estrarre dati preziosi.

Test interni. In un test interno, un tester con accesso a un'applicazione dietro il firewall simula un attacco da parte di un insider malintenzionato, ad esempio in seguito al furto di credenziali interne all'organizzazione a causa di un attacco di phishing.

Blind test. In un blind test, un tester dispone solo di poche informazioni essenziali, simulando quindi quello che avverrebbe durante un reale assalto all'infrastruttura. In questo caso però, il personale addetto alla sicurezza informatica è a conoscenza dell'imminente attacco, predisponendo quindi sia i necessari livelli di sicurezza, sia i sistemi di logging atti all'analisi real-time dei sistemi in testing.

Double-blind testing. In un double-blind test, il personale di sicurezza non è a conoscenza dell'attacco.

Test mirati. In questo scenario, sia il tester che il personale di sicurezza lavorano in sincrono elaborando quindi informazioni mirate.

2.4 Modalità e strumenti per l'esecuzione del testing della piattaforma

L'implementazione delle suite di testing della piattaforma è stata condotta in buona parte di pari passo all'implementazione della piattaforma stessa, soprattutto riguardo ai test di unità, coerentemente con le best practices dettate dell'Ingegneria del Software.

Inoltre, come descritto con maggiore dettaglio nella Sezione 4, nel periodo subito precedente al testing dei servizi applicativi prototipali, è stata pianificata una fase di potenziamento, raffinamento e refactoring dei test, al fine di garantire una validazione efficace dei servizi applicativi prototipali sui quali effettuare, appunto, la sperimentazione in vitro presso gli utenti finali: come da diagramma GANTT in Figura 2, nella parte finale del mese di Luglio 2019 e durante tutto il mese di Settembre 2019 è stata pianificata l'implementazione avanzata delle tre categorie di test precedentemente descritte.

2.4.1 Strumenti selezionati

Nella pianificazione della sperimentazione e a supporto degli sviluppatori, sono stati proposti degli strumenti utili all'implementazione dei test, tenendo presente che la realizzazione della piattaforma si è basata su ambiente di sviluppo .NET e linguaggio di programmazione C#:

Unit Test. La tecnologia selezionata per condurre gli Unit Test è *NUnit*, unit testing framework open source per Microsoft .NET [2]. NUnit appartiene alla famiglia Xunit e segue gli stessi propositi di JUnit per il mondo Java.

Testing Strutturale. Per soddisfare le esigenze di implementazione di un sistema di testing strutturale automatizzato, si è fatto affidamento a specifici strumenti software, tra cui Pex (Program EXploration), uno strumento per la generazione e gestione automatizzata di test white-box che fornisce risultati approfonditi ed affidabili in ambienti .NET [3].

Penetration Test. La selezione di tools, tecniche e metodi per l'esecuzione automatizzata e semi-automatizzata dei Penetration test si è basata sulle direttive dettate dalla community Open Web Application Security Project (OWASP)[5], l'organizzazione di riferimento dell'application security. L'organizzazione traccia periodicamente una serie di direttive e indicazioni utili all'implementazione di valide application security policy, raccogliendole in un documento distribuito pubblicamente, il **OWASP Top 10 web application security risks**. La verisone attualmente più aggiornata è l'OWASP Top 10 2017 [6].

In quest'ottica sono stati selezionati i seguenti tool per impostare ed eseguire l'ambiente di test:

- Vooki: fornito da VegaBird Technologies e conforme agli standard OWASP TOP 10. Mediante Vooki è possibile sottoporre a Penetration test automatici sia applicazioni web, sia REST API.
- Postman: è il REST API client utilizzato durante l'implementazione della piattaforma, il quale, inoltre, permette di eseguire script nelle chiamate agli endpoint nelle varie fasi di esecuzione delle stesse. Per tali ragioni l'ambiente di testing per le REST API è stato potenziato mediante l'esecuzione di script implementati ad-hoc.

3. Caratterizzazione del testing dei servizi applicativi prototipali

Il testing dei servizi applicativi prototipali mira a stabilire l'usabilità dei dimostratori web e mobile ottenuti al termine della fase implementativa del progetto.

La misurazione dell'usabilità si basa sull'analisi dei dati relativi all'interazione utente-prodotto. Per ottenere una validità statistica dei dati è necessario definire con attenzione sia gli obiettivi di usabilità sia i compiti da eseguire e le relative tecniche di valutazione delle prestazioni: da questo nasce l'esigenza di affidarsi al metodo MiLE+, descritto nel seguito.

3.1 Il metodo MiLE+

Per la valutazione delle metriche di usabilità ci si è basati sul metodo MiLE+ (MIlano-Lugano Evaluation method), un framework per la valutazione dell'usabilità delle applicazioni software che mira all'analisi di tutti gli elementi coinvolti nell'utilizzo dell'applicazione: la grafica, la tecnologia, la struttura navigazionale, i contenuti, alcuni aspetti cognitivi dell'interfaccia .

Il metodo garantisce un buon equilibrio tra *valutazioni euristiche* e *tecniche task-driven*, cioè basate su determinati task che un generico utente dovrebbe eseguire nello svolgimento degli scenari di utilizzo imposti dai requisiti dell'applicazione[7][8].

A questo proposito, MiLE+ propone due tipologie di ispezione dell'applicazione:

1. **Technical Inspection**: valuta la qualità del design e mette in luce i difetti implementativi. È una tipologia di ispezione "astratta", poiché non necessariamente deve essere legata agli scenari di utilizzo tipici dell'applicazione su cui si applica.

Si basa su euristiche che quantificano tecnicamente l'usabilità in base a parametri concernenti la navigabilità, i contenuti, la tecnologia e il design dell'interfaccia;

2. **User Experience Inspection**: riguarda quegli aspetti dell'usabilità fortemente legati all'utilizzo dell'applicazione da parte dell'utente.
   In tale ispezione, a partire da ogni requisito funzionale dell'applicazione, vengono generati degli scenari di utilizzo della stessa ed ognuno di essi viene suddiviso in diversi task. Ogni task viene sottoposto all'analisi di usabilità per quantificarne la qualità dell'interazione tra l'utente e i contenuti dell'applicazione, la qualità della navigabilità e quanto i suoi aspetti cognitivi siano coerenti con quelli dell'utente e, infine, in che misura l'interazione con l'applicazione sia apprezzata dall'utente.

   In altri termini, la User Experience Inspection permette di conoscere in anticipo i potenziali problemi che potrebbero insorgere nel momento in cui un utente utilizza l'applicazione.

Gli **scenari applicativi** sui quali si basano le misurazioni dell'usabilità dell'applicazione rappresentano *i casi d'uso derivanti dai requisiti funzionali definiti in fase progettuale*. Essi consistono di tre elementi e possono essere rappresentati da tabelle in cui vengono definiti:

- Il **profilo dell'utente** al quale si riferisce lo scenario;
- lo **scopo finale** dello scenario;
- i **task** in cui lo scenario è suddiviso, tali da permettere all'utente di ottenere lo scopo dichiarato.

Un esempio di scenario per la valutazione dell'usabilità di un'applicazione di e-learning è rappresentato nella seguente tabella:

Tabella 1 – Esempio di scenario MiLE+

USER PROFILE	Marc, 26 anni, vorrebbe frequentare un corso on-line. Sebbene utilizzi frequentemente Internet, non ha mai usato una piattaforma di e-learning
GOAL	Conoscere i corsi on-line disponibili
TASKS	• Accedere all'elenco di corsi • Conoscere la struttura di ogni corso • Ottenere i contatti dei tutors dei corsi

3.1.1 Technical Inspection

Come già affermato, la Technical inspection ha lo scopo di identificare i problemi di design ed i difetti implementativi. Durante l'analisi viene considerato il punto di vista del designer, e non quello dell'utente finale (a differenza di quanto avviene nella User Experience Inspection).

Di seguito vengono valutate le dimensioni del design sulle quali si appoggia la Technical Inspection e di ognuna viene riportato l'elenco delle diverse euristiche utilizzate:

Content. qualità dei contenuti, in termini di efficacia di comunicazione. È composta dalle seguenti euristiche:

Tabella 2 - MiLE+ Technical Inspection: euristiche Content

Accuracy	L'accuratezza stabilisce se il contenuto testuale descrive adeguatamente i temi ai quali si riferisce, nonché se è consistente in se stesso
Currency	I contenuti devono essere aggiornati il più possibile al periodo al quale l'utente vuole riferirsi. Ad ogni modo deve essere possibile stabilire chiaramente la loro data di pubblicazione e il loro periodo di validità
Coverage	Devono essere chiari gli argomenti ai quali si riferisce il testo ed i limiti di copertura di tali argomenti da parte dei contenuti
Content objectivity	Deve essere possibile stabilire chiaramente l'oggettività dell'autore nei confronti dei contenuti da lui riportati. Può essere importante capire se l'autore ha interessi particolari a riportare determinati contenuti (per esempio, nel caso di pubblicità)
Authority	L'autorità dell'autore può essere vista sotto due aspetti: competenza dell'autore nei confronti dei contenuti riportati e buona predisposizione verso i lettori
Conciseness	Livello di sinteticità e concisione del testo
Text errors	Nel testo non dovrebbero essere presenti errori grammaticali
Multimedia consistency	I contenuti multimediali devono essere consistenti con gli argomenti trattati nella pagina in cui tali contenuti sono riportati

Navigation. la dimensione della navigabilità può essere indagata sotto due differenti punti di vista: i diversi modi in cui un utente può raggiungere specifiche porzioni di informazione e le connessioni tramite cui passare da un contenuto ad un altro.

Tabella 3 - MiLE+ Technical Inspection: euristiche Navigation

Segmentation	Differenti informazioni in merito ad uno stesso topic potrebbero essere suddivise, segmentate, in diverse viste dell'applicazione. Risulta importante che l'utente possa comprendere quali viste appartengano ad uno stesso topic e come la navigazione tra queste viste funzioni
Orientation clues	Durante la navigazione di un particolare topic è molto importante che un utente capisca immediatamente la sua posizione all'interno

	del topic
Accessibility of different views	è sempre essenziale che tutte le viste di un topic siano accessibili in pochi click
Landmarks	Rappresenta quanto l'accesso alle varie sezioni dell'applicazione sia facilitato dalla presenza di landmarks. Euristica utile soprattutto nel caso di applicazioni web
Consistency	Tutte le applicazioni hanno in generale un'architettura navigazionale. La navigabilità tra le differenti parti dell'applicazione, basata su tale architettura, deve esse quanto più consistente è possibile. Questo garantisce la rapida comprensione dell'utente di come la navigazione tra le viste funzioni.

Technology/Preformance. la dimensione della tecnologia e delle performance di un'applicazione software riguarda quegli aspetti concernenti la correttezza formale del codice, le gestione delle sezioni critiche e la reazione del sistema in caso di errori o comportamenti inattesi dell'utente.

Tabella 4 - MiLE+ Technical Inspection: euristiche Technology/Preformance

System reaction to errors of a user	La gestione efficiente degli errori rappresenta un punto cardine dell'usabilità di un sistema
Scripting errors	L'euristica riguarda gli errori a run-time generati dai sistemi costruiti con linguaggi di scripting
Client-side code interpretation compatibility	Nel caso di applicazioni web, il codice lato client è notoriamente non supportato ed interpretato allo stesso modo da tutti i browser. L'implementazione deve sempre tenerne conto
Page download time	Nelle applicazioni web, il tempo di caricamento di una pagina non deve eccedere determinati limiti
Media streaming	Nelle applicazioni web, lo streaming di contenuti multimediali dovrebbe essere ottimizzato per connessioni lente

Interface design.

Tabella 5 - MiLE+ Technical Inspection: euristiche Interface Design

Euristiche semiotiche riguardano quelle euristiche che quantificano quanto sia facile per un utente comprendere il messaggio proposto durante l'interazione dell'utente con l'applicazione	
Ambiguity / Clarity	I termini usati dall'applicazione potrebbero essere interpretati con significati differenti da parte dell'utente
Information Scent	Al di là dei contenuti testuali, all'utente potrebbero essere mostrate differenti tipologie di contenuti addizionali, per facilitare le sue scelte nell'utilizzo dell'applicazione (thumbnail, suoni, ecc..)
Grouping adequacy	Indica quanto sono adeguati i raggruppamenti contestuali dei contenuti nelle differenti viste
Position of importance	Ogni vista ha un particolare scopo riferito ad un particolare topic. Riconoscere entrambi è legato ad un adeguato posizionamento spaziale degli elementi nelle viste
Euristiche cognitive	
Information overload	Ogni vista è composta da un insieme di differenti messaggi, ognuno avente un particolare significato. Quantità eccessive di messaggi in un'unica vista e alti gradi di eterogeneità potrebbero richiedere eccessivi sforzi da parte di utenti che si interfacciano per la prima volta con l'applicazione
Scannability	Le viste devono essere il più possibile "scannables", cioè la loro struttura deve essere ordinata in base ad elementi facilmente individuabili da un utente (un link, un testo, un'immagine, ecc.), il quale può, in tal modo, individuare facilmente le aree di maggiore interesse
Euristiche grafiche Riguardano due aspetti: il design grafico, riferito alle scelte di colori, tipologia di caratteri, icone e altri elementi grafici, ed il layout, cioè la distribuzione spaziale degli elementi all'interno delle viste	

Use of chromatic code	Identifica la correttezza nell'uso di differenti colorazioni per identificare sezioni e sottosezioni delle viste, attrarre l'attenzione dell'utente, ecc.
Background contrast	Il colore dello sfondo deve sempre consentire la facile lettura dei contenuti della vista
Font size/colour/type	Il testo deve essere sempre facilmente leggibile ed evidenziare i ruoli delle differenti sezioni dei contenuti. La dimensione, la tipologia di caratteri e il colore del testo devono rispettare questi requisiti
Text layout	Dividere in più parti un testo troppo lungo ne migliora la leggibilità. Lo stesso vale per un corretto uso della giustificazione
Anchor identity/states	Stabilisce quanto sia facile individuare le ancore (link, pulsanti, ecc.) nelle viste. Inoltre, è importante identificare i diversi stati in cui tali ancore si trovano (link visitati, cursore del mouse nel focus di un pulsante, ecc.)
Layout grid consistency	La struttura del layout deve essere consistente
Layout convention	Gli utenti dei paesi occidentali sono nel percepire i contenuti informativi sono vincolati da condizioni quali: • Le viste vanno interpretate da sinistra a destra, dall'alto verso il basso; • Gli items di dimensioni maggiori sono i più rilevanti; • Ecc..

Un breve esempio di Technical Inspection è mostrato nella seguente tabella:

Tabella 6 – Esempio di Technical Inspection in MiLE+

Dimensione	Euristica	Punteggio	Commento
Content	Accurancy	3	Testo troppo lungo e difficile da leggere
	Text errors	9	Il testo non presenta errori
Navigation	Accessibility of different views	9	Tutte le pagine sono facilmente accessibili
	...		

A seconda delle esigenze, è possibile legare ogni ispezione ad un particolare scenario, anche se nella Technical Inspection ciò non rappresenta un requisito fondamentale.

3.1.2 User Experience Inspection

Questa tipologia di ispezione è da considerarsi *scenario-based*, poiché ogni valutazione dell'usabilità è da legare necessariamente ad un particolare scenario applicativo.

Durante la User Experience Inspection, l'esaminatore deve "mettersi nei panni dell'utente" e valutare i parametri di usabilità in base a particolari criteri ed euristiche molto vicine al punto di vista dell'utente finale, denominate *User Experience Indicators* (UEIs), divise in tre categorie principali:

Content Experience Indicators. misura della qualità dell'interazione dell'utente con i contenuti dell'applicazione

Tabella 7 – MiLE+ User Experience Inspection: Content Experience Indicators

Completeness	L'utente riesce a trovare tutte le informazioni richieste
Richness	Questo UEI si riferisce alla quantità di informazioni che chiariscono il contenuto
Comprehensibility	La comprensibilità è legata alla capacità del contenuto di essere auto esplicativo. In altri termini, il topic principale dovrebbe essere chiaro e non ambiguo
Relevance	Le informazioni esplicitate nei contenuti devono corrispondere ai bisogni degli utenti che si interfacciano con l'applicazione
Multilinguisticity	le applicazioni software, specialmente se web-based,

	dovrebbero essere multilingua
Multimediality	La multimedialità, se consistente con i topics di un'applicazione, garantisce un maggiore trasporto dell'informazione
Satinsfaction	La capacità di un contenuto di soddisfare un utente indica che le informazioni riportate raggiungono i desideri, i bisogni e gli scopi degli utenti

Navigation & Cognitive Experience Indicators. viene valutato quanto la navigazione sia funzionale e quanto gli aspetti cognitivi della stessa siano conformi a quelli dell'ambiente tipico dell'utente

Tabella 8 - MiLE+ User Experience Inspection: Navigation & Cognitive Experience Indicators

Self-evidence	È una proprietà riguardante gli elementi interattivi (simboli, icone, link testuali, bottoni, immagini, ecc.) utilizzati a supporto di alcune operazioni elementari. Tali elementi dovrebbero essere auto esplicativi, non ambigui e consistenti con la semantica dell'operazione
Predictability	È la capacità degli elementi interattivi di anticipare il contenuto correlato con essi
Learnability	Rappresenta la capacità dell'applicazione di essere chiara per l'utente, con riferimento alla navigabilità, alla strategia visuale degli elementi interattivi, alla creazione di un mappa mentale della struttura, ecc.
Information overload	Si riferisce alla quantità di messaggi e alla loro eterogeneità
Accessibility	Il contenuto dovrebbe essere letto e gestito da chiunque, a prescindere dall'esperienza, dalla sua localizzazione e dalla tecnologia software utilizzata
Memorability	Si riferisce alla capacità mentale di ricordare precedenti esperienze riguardo all'applicazione. Nel momento in cui un utente ritorna ad utilizzare l'applicazione dopo un certo periodo di tempo, dovrebbe essere in grado di ristabilire velocemente un'alta capacità di utilizzo della stessa

Interaction Flow Experience Indicators. misura quanto l'utilizzo dell'applicazione sia realmente apprezzata dall'utente.

Tabella 9 - MiLE+ User Experience Inspection: Interaction Flow Experience Indicators

Naturalness	L'applicazione dovrebbe presentare una semantica facile da comprendere, utilizzabile, quindi, con naturalità da un utente (ad esempio, un immagine rappresentante una casa dovrebbe essere usata esclusivamente come link alla home page)
Effectiveness	L'efficacia rappresenta la capacità di un utente di conseguire il proprio scopo, misurata in base alla frequenza di successi in ogni task, al numero di ritorni e al tempo impiegato per compiere i task
Engagement	Rappresenta la capacità del sistema di rapire ed estasiare l'utente, grazie alla qualità sia dei contenuti, sia dell'interazione complessiva offerta
Precision	La precisione misura la congruenza semantica tra le informazioni richieste e le risposte ottenute dal sistema. In tal senso, essa rappresenta la capacità del sistema di fornire informazioni puntuali e non di sovraccaricare l'utente con informazioni ridondanti e non desiderate
Satisfaction of the experience	La soddisfazione dell'utente significa, in generale, che ha raggiunto i suoi scopi facilmente. Può essere vista come un macro indicatore dell'esperienza dell'utente nell'utilizzo dell'applicazione, che racchiude in se buona parte degli indicatori visti precedentemente

Un esempio di utilizzo delle UEIs è mostrato nelle seguenti tabelle, in cui uno specifico utente deve navigare tra le pagine del sito web del museo del Louvre per ottenere maggiori informazioni sul di esso:

Tabella 10 – MiLE+: scenario di esempio

SCENARIO	
USER PROFILE	Appassionato d'arte
GOAL	Ottenere maggiori informazioni sul museo del Louvre
TASKS	• Ottenere informazioni generali sul museo • Conoscere le maggiori opere esposte • Conoscere la struttura del museo • Ottenere informazioni sulla storia del museo

Per ogni task indicato, è necessario valutare la varie euristiche descritte precedentemente.

Ad esempio, riguardo al solo task "Ottenere informazioni sulla storia del museo":

Tabella 11 – Esempio di schema di valutazione della UEI per uno specifico task

	UEIs				
Task: Ottenere maggiori informazioni sul museo del Louvre	**Predictability**	**Understandability**	**Richness**	**Comprehensibility**	**Punteggio totale per il task**
Punteggi	8	8	5	6	6.75
Pesi	0.1	0.1	0.5	0.3	
Punteggi pesati	0.8	0.8	2.5	1.8	**5.9** (media pesata)

Come è possibile vedere, ad ogni UEI viene associato un peso, che ne identifica l'importanza relativamente al task preso in esame.

Nel complesso, l'applicazione sarà usabile dal punto di vista dell'utente, riguardo al task preso in esame, nella misura indicata dalla media pesata calcolata.

In generale, il numero di euristiche previsto dal framework MiLE+ è volutamente ampio, per garantirne la compatibilità con il maggior numero possibile di applicazioni software. Per questo motivo, per quanto riguarda sia la Technical Inspection, sia la User Experience Inspection, le euristiche da selezionare nelle valutazioni deve essere ristretto di volta in volta a quelle maggiormente indicative nel caso specifico preso in esame.

3.2 Modalità di svolgimento del testing dei servizi applicativi

Gli obiettivi da perseguire nella fase di sperimentazione dei servizi applicativi prototipali sono quelli di validare e valutare i servizi utente della piattaforma EasyPAL esposta su piattaforme Web e Mobile, mediante una sperimentazione "guidata" e ben strutturata. Come già definito in precedenza, la sperimentazione sarà condotta da parte di 2 macro-categorie di utenti:

- Cittadino (almeno n. 75)
- Operatore comunale: Responsabili IT e dirigenti; Funzionari amministrativi (almeno n. 10); Rappresentanti delle associazioni di categoria (almeno n. 20);

I servizi da sperimentare sono dati da:

- Anagrafe: visualizzazione dati anagrafici utente e richiesta certificati online mediante ANPR.
- Pagamento: pagamento online mediante PagoPA di tributi comunali, sanzioni amministrative e al codice della strada, ecc.

- Documentazione: visualizzazione/download documentazione amministrativa dell'utente (dichiarazioni, avvisi di pagamento, avvisi di accertamento, ingiunzioni ecc.)
- Account: funzionalità di login mediante SPID

La differenza fra gli utenti Cittadino e Operatore comunale consiste nel fatto che, mentre il primo può gestire i servizi relativi solo alla propria anagrafica, il secondo può gestire gli stessi servizi relativamente a tutte le anagrafiche del Comune di riferimento, attraverso ricerche mirate e basate sul codice fiscale.

La fase di sperimentazione dei servizi applicativi verrà avviata da una serie di incontri con i Municipi coinvolti, Altamura e Laterza innanzitutto, durante i quali verrà distribuito il link alla pagina Web di progetto con i servizi da testare.

Durante gli incontri verrà, inoltre, redatta una lista degli indirizzi email dei partecipanti: in tal modo gli stessi partecipanti riceveranno per email un link attraverso il quale scaricare ed installare l'applicazione Mobile per sistemi Android: non essendo ancora in produzione, l'applicazione non sarà caricata sugli application store ufficiali, ma verrà caricata sui server di progetto e sarà accessibile, appunto, mediante URL dedicato.

Una volta testate e sperimentate le funzionalità, gli utenti potranno esprimere le proprie opinioni attraverso gli strumenti messi a disposizione dal framework Moduli Google, ovvero gli strumenti forniti da Google per creare moduli e sondaggi. I vantaggi dell'utilizzo di questa piattaforma sono molteplici e riassumibili nei seguenti punti:

- è possibile rappresentare in formato semplice e intuitivo le informazioni riportate nelle tabelle MiLE+ redatte per gli scenari da testare;
- i risultati dei test sono facilmente esportabili ed elaborabili;
- la piattaforma è liberamente accessibile e non necessita di ulteriore infrastruttura hardware e software.

3.3 Implementazione del metodo MiLE+

Di seguito vengono descritti gli scenari da testare utilizzando il metodo MiLE+. Gli scenari sono stati analizzati tenendo presente le diverse tipologie di utente ed i servizi ad essi associati, descritti nel precedete paragrafo.

3.3.1 Descrizione degli scenari MiLE+

Di seguito sono riportate le tabelle che descrivono gli scenari applicativi individuati, da sottoporre a test di usabilità mediante MiLE+.

Tabella 12 – Scenario Cittadino/funzionalità Anagrafe

SCENARIO	
USER PROFILE	Cittadino
GOAL	Visualizzazione dati anagrafici utente e richiesta certificati online mediante ANPR
TASKS	• Autenticazione al sistema • Accesso alla sezione Anagrafica • Visualizzazione dati anagrafici • Richiesta certificati online mediante ANPR

Tabella 13 - Scenario Cittadino – funzionalità Pagamento servizi

SCENARIO	
USER PROFILE	Cittadino
GOAL	Pagamento servizi online
TASKS	• Autenticazione al sistema • Accesso alla sezione Pagamento • Visualizzazione delle voci relative ai pagamento da effettuare • Avvio della procedura di pagamento PagoPA

Tabella 14 - Scenario Cittadino – funzionalità Documentazione

SCENARIO	
USER PROFILE	Cittadino
GOAL	Gestione documentazione utente
TASKS	• Autenticazione al sistema • Accesso alla sezione Documentazione • Visualizzazione e download della propria documentazione amministrativa

Tabella 15 - Scenario Operatore comunale – funzionalità Anagrafe

SCENARIO	
USER PROFILE	Operatore comunale
GOAL	Visualizzazione dati anagrafici di uno specifico cittadino e richiesta

	certificati online mediante ANPR
TASKS	• Autenticazione al sistema • Accesso alla sezione Anagrafica • Ricerca delle informazioni mediante il codice fiscale del cittadino • Visualizzazione dati anagrafici • Richiesta certificati online mediante ANPR

Tabella 16 - Scenario Operatore comunale – funzionalità Documentazione

SCENARIO	
USER PROFILE	Operatore comunale
GOAL	Gestione documentazione di uno specifico cittadino
TASKS	• Autenticazione al sistema • Accesso alla sezione Documentazione • Ricerca delle informazioni mediante il codice fiscale del cittadino • Visualizzazione e download della propria documentazione amministrativa

3.3.2 Valutazione della User Experience

Tabella 17 – Schema valutazione UEI task "Accesso alla pagina di accesso ed avvio procedura di Log-in mediante SPID"

	UEIs				
Tasks: Accesso alla pagina di accesso ed avvio procedura di Log-in mediante SPID	**Self-evidence**	**Comprehensibility**	**Accessibility**	**Memorability**	**Punteggio totale per il task**
Punteggi					
Pesi	0.2	0.4	0.3	0.1	
Punteggi pesati					

Tabella 18 - Schema valutazione UEI task “Visualizzazione dati anagrafici utente e richiesta certificati online mediante ANPR”

		UEIs				
Tasks: Visualizzazione dati anagrafici utente e richiesta certificati online mediante ANPR	**Completeness**	**Comprehensibility**	**Richness**	**Learnability**	**Predictability**	**Punteggio totale per il task**
Punteggi						
Pesi	0.2	0.2	0.2	0.3	01	
Punteggi pesati						

Tabella 19 - Schema valutazione UEI task “Visualizzazione delle voci relative ai pagamento da effettuare ed avvio della procedura di pagamento PagoPA”

		UEIs		
Tasks: Visualizzazione delle voci relative ai pagamento da effettuare ed avvio della procedura di pagamento PagoPA	**Self-evidence**	**Comprehensibility**	**Precision**	**Punteggio totale per il task**
Punteggi				
Pesi	0.2	0.2	0.2	
Punteggi pesati				

Tabella 20 - Schema valutazione UEI task “Visualizzazione e download della propria documentazione amministrativa”

		UEIs				
Task: Visualizzazione e download della propria documentazione amministrativa	**Completeness**	**Comprehensibility**	**Richness**	**Effectiveness**	**Predictability**	**Punteggio totale per il task**
Punteggi						
Pesi	0.2	0.2	0.2	0.2	0.2	
Punteggi pesati						

Tabella 21 - Schema valutazione UEI task “Ricerca del dati del cittadino mediante codice fiscale”

		UEIs				
Task: Ricerca del dati del cittadino mediante codice fiscale	**Completeness**	**Comprehensibility**	**Richness**	**Learnability**	**Predictability**	**Punteggio totale per il task**
Punteggi						6.75
Pesi	0.2	0.2	0.2	0.3	01	
Punteggi pesati						**5.9** (media pesata)

Tabella 22 – Schema valutazione Technical Inspection

Dimensione	Euristica	Punteggio
Content	Accurancy	
	Currency	
Navigation	Accessibility of different views	
	Segmentation	
	Consistency	
Technology/Preformance	System reaction to errors of a user	
	Page download time	
Interface design	Ambiguity / Clarity	

4. Pianificazione temporale della sperimentazione

La pianificazione temporale delle attività è stata elaborata sulla base delle fasi di sperimentazione da implementare e descritte nei capitoli precedenti.

Il seguente diagramma GANTT mostra la programmazione delle attività di sperimentazione.

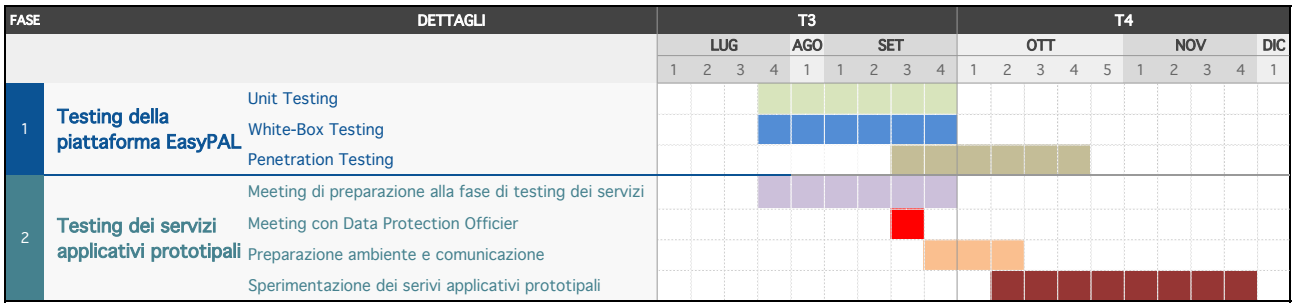

Figura 2 - diagramma GANTT del piano di sperimentazione

Dal diagramma si evince che le attività relative al testing della piattaforma sono state poste in anticipo rispetto a quelle di sperimentazione dei servizi prototipali, in quanto propedeutiche alla realizzazione di prototipi stabili e robusti.

Inoltre, contemporaneamente alla fase di testing della piattaforma è stata schedulata una fase di preparazione dell'ambiente di test e delle strategie di comunicazione relativamente al testing dei servizi applicativi prototipali. Lo scopo di questa fase preliminare è quello di ospitare le applicazioni web e mobile e comunicare ai partecipanti alla sperimentazione le modalità di svolgimento della stessa.

Sempre da GANTT, come premessa alla sperimentazione dei servizi prototipali sono previsti gli incontri con i Municipi coinvolti e con gli esperti legali per garantire conformità della sperimentazione con i Regolamenti europei in materia di Tutela della Privacy.

Riferimenti

[1] H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 366–427, 1997.

[2] A. Hunt, D. Thomas, and Pragmatic Programmers (Firm), *Pragmatic unit testing : in C# with NUnit*. Pragmatic Bookshelf, 2004.

[3] N. Tillmann and J. De Halleux, "Pex-white box test generation for .NET," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 4966 LNCS, pp. 134–153.

[4] H. H. Thompson, "Application penetration testing," *IEEE Security and Privacy*, vol. 3, no. 1. pp. 66–69, Jan-2005.

[5] "OWASP." [Online]. Available: https://www.owasp.org/index.php/Main_Page. [Accessed: 02-Oct-2019].

[6] "OWASP Top 10-2017," 2003.

[7] D. Bolchini, "MiLE+ (Milano-Lugano Evaluation method) A systematic approach to usability evaluation," 2004.

[8] L. Triacca, D. Bolchini, L. Botturi, and A. Inversini, "MiLE: Systematic usability evaluation for e-learning web applications," in *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*, 2004, pp. 4398–4405.