futura-dev-guide Documentation

Release 1

Marco Buccio

Jan 18, 2024

CONTENTS

Il presente manuale descrive la struttura dell'applicazione Futura nei suoi componenti principali, il framework di riferimento e le guide linea che risulta opportuno rispettare nell'implementazione di estensioni.

CHAPTER

ONE

STRUTTURA DELL'APPLICAZIONE

L'applicazione Futura si basa su un Framework sviluppato da Marco Buccio <info@mbuccio.it> per lo sviluppo di applicazioni web basate su PHP. Tale framework nasce con l'obiettivo di strutturare, uniformare e velocizzare lo sviluppo di applicazioni web completamente customizzate.

1.1 Componenti

Lo sviluppo applicativo avviene rispettando la seguente struttura:

- action: sono le classi PHP che si occupano di gestire l'interazione con l'utente e la generazione dell'output sia esso costituito da codice HTML o da risposte ad API REST;
- constant: contiene tutte le definizioni trasversali e statiche al sistema;
- css: fogli di stile per la formattazione delle pagine HTML generate;
- env: contiene la configurazione dell'ambiente. si veda la guida per l'installazione e la configurazione.
- filter: filtri delle richieste, sono classi utilizzate per gestire aspetti trasversali alle richieste. Ad esempio, il LoginFilter controlla i diritti di accesso di un utente;
- img: risorse statiche;
- lib: librerie utilizzate dall'applicazione;
- script: script JavaScript utilizzati dal frontend;
- service: classi di servizio che implementano logiche di business e accesso ai dati;
- template: template per la produzione di documenti e messaggi email;
- ui: componenti dell'interfaccia grafica;
- utils: classi di utilità;
- Scheduler.php: punto di accesso per i tasks schedulati;
- index.php: punto di accesso per le richieste HTTP sia per le richieste di pagine HTML sia per l'interrogazione di API interne;

1.2 Flusso delle richieste

La'plicazione presenta due principali punti di accesso delle richieste

1.2.1 Attività schedulate

Queste attività possono essere richiamate solo localmente al sistema direttamente da linea di comando o inserendo l'opportuna schedulazione nella crontab. Si veda la guida di installazione e configurazione. Lo scheduler è costituito da una classe che si occupa di rendere disponibili i componenti del framework, attivare la connessione ai dati ed invocare l'opportuno metodo, solitamente inserito in una classe di servizio. La classe scheduler non gestisce la periodicità e la ciclicità dell'invocazione di una chiamata ma solo il rooting applicativo.

1.2.2 Richieste HTTP

Le richieste HTTP transitano dall'index.php attraverso la configurazione del mod-rewrite di Apache. L'utilizzo di questo approccio rende uniforme la gestione di tutte le richieste e garantisce un ciclo di richiesta controllato. I passi principali eseguiti nel processo sono:

- Attivazione dei componenti del framework;
- Connessione al database;
- Parsing della richiesta;
- Istanziazione della classe di riferimento predisposta per la gestione della richiesta;
- attivazione dei filtri di gestione, ad esempio per il controllo degli accessi e per la gestione dei parametri in ingresso;
- Esecuzione del metodo richiesto;
- restituzione dei risultati generati;

1.3 Estensioni alla gestione del frontend

Nelle ultime iterazioni di implementazione della piattaforma è stato introdotto l'utilizzo del framework per lo sviluppo di SPA: VueJS. Il framework consente di integrare un approccio di sviluppo del frontend più moderno ad applicazioni web senza la necessità di passare in toto all'approccio previsto per altri sistemi SPA. In Futura, infatti, VueJS viene usato per la creazione di componenti specifici, ad esempio le tabelle o i test della sezione Mondo delle Parole. A tendere tutto il frontend dovrebbe essere realizzato con VueJs lasciando lato server la sola fornitura di dati tramite API REST. Per eseguire la preparazione dei componenti VueJS è necessario eseguire il comando:

` nom run build `

Questo genera gli script presenti nella cartella public-html/scripts/mb/vue-components che possono essere rilasciati nei vari ambienti.

1.4 Docker

Per consentire di attivare rapidamente l'ambiente di sviluppo è stato predisposta la configurazione dell'ambiente tramite docker-compose. Per avviare l'ambiente in locale è necessario installare Docker desktop ed eseguire il comando:

` docker-compose up `

CHAPTER

TWO

SICUREZZA APPLICATIVA

La sicurezza applicativa nell'acesso alle risorse del frontend è fondamentale per tutte le applicazioni web. L'applicazione basa la gestione della sicurezza su una serie di meccanismi:

- Gestione centralizzata delle richieste: l'impiego di mod-rewrite consente di centrallizzare e uniformare la procedura di accesso;
- Separazione delle funzionalità in packages: le funzionalità applicative vengono suddivise in packages fortemente vincolati ai permessi dell'utente;
- Strutturazione gerarchica delle Action;
- Accesso gestito ai dati: l'accesso ai dati avviene con query eseguite tramite PDO (PHP Data Object) che permette di limitare i rischi di attacchi;
- L'esposizione di form pubblici viene protetto dalla presenza dell'integrazione con il sistema Google re-CAPTCHA per l'individuazione di bot e comportamenti malevoli.
- È stato integrato il login degli utenti tramite Azure AD per consentire un processo di login standardizzato e maggiormente sicuro.

A tali approcci si aggiunge ovviamente una gestione del sistema adeguata e l'accesso all'intero sistema basato esclusivamente su connessioni sicure.

2.1 Packages

I principali packages di cui è costituita l'applicazione sono:

- **admin: contiene tutte le interazioni di tipo amministrativo. I sotto packages sono**
  - data: informazioni sui dati;
  - dropOut: informazioni amministrative legate al modulo di Drop Out;
  - edSalute: pannello amministrativo legato ai progetti di Educazione alla Salute;
  - eModel: pannello amministrativo del modulo Modello E;
  - pei: pannello amministrativo modulo PEI-PDP;
  - **portfolio: pannello amministrativo legato al modulo Portfolio Docenti;**
    - technical: accesso a configurazioni tecniche;
    - user: gestione degli utenti;
- api: accesso alle API di integrazione;
- icf: albero dei contenuti ICF-CY;
- public: tutte le azioni pubbliche tra cui la pagina di login;
- **structure: azioni con limite di accesso ai membri scolastici;**
  - dropOut: azioni di gestione a livello di struttura del modulo Drop Out;

  - edSalute: azioni di gestione a livello di struttura del modulo di gestione dei progetti di Educazione alla Salute;
  - pdp: gestione PDP a livello di struttura;
  - pei: gestione del PEI a livello di struttura;
  - portfolio: gestione del Portfolio Docenti a livello di struttura;
  - user: gestione dati studente a livello di struttura;
  - pei_tirocinio: gestione del PEI di tirocinio;
  - eModel: ambito del modello E;
  - test: legato al progetto Mondo delle Parole, progetto Letto Scrittura;
  - servizio_valutazione: attiva l'integrazione con il progetto Servizio di Valutazione;
- taxonomy: accesso alla tassonomia del PEI e del PDP;
- user: azioni specifiche dell'utente ad esempio la gesitone del profilo personale;

I ruoli degli utenti sono direttamente legati ad alcuni packages per cui, ad esempio, per un utente con ruolo insegnante sarà impossibile accedere alla sezione admin dato che il ruolo limita l'accesso al solo package structure. La definizione dei ruoli è dinamica e risulta modificabile dagli amministratori del sistema che abbiano accesso al package admin/user. Si veda la guida utente per gi amministratori.

2.2 Accesso ai dati

Per garantire un approccio logico pulito e consistente l'accesso ai dati viene gestito in modo centrallizzato dall'EntityManager (EM) che fa uso di PDO (PHP Data Object). È preferibile evitare l'uso dell'EM direttamente dalle classi di action che vengono istanziate in funzione della necessità della richiesta relegando l'accesso alle classi Service. Ad esempio, ll PeiService conterrà tutte i metodi utilizzati per la gestione del PEI in termini di dati. Ad esempio il metodo PeiService::create, permette di creare un nuovo PEI per uno specifico utente:

```
public static function create($userYearId, $createdByUserId, $type){ $peiId =
EM::insertEntity("user_year_pei",
    [
        'user_year_id' => $userYearId, 'creation_date' => 'NOW()', 'cre-
        ated_by_user_id' => $createdByUserId, 'type' => $type
    ]);
return $peiId;
}
```

2.3 Accessi utente

Tutti gli accessi da parte degli utenti vengono salvati in un'apposita tabella e sono visibili dal pannello amministrativo. Inoltre, dal pannello amministrativo è possibile monitorare gli utenti attualmente loggati.

Il sistema presenta agli utenti che eseguono il login le informazioni relative all'ultimo accesso eseguito (data e ora). L'utente può così monitorare eventuali abusi avvenuti sul proprio account richiedendo la chiusura dello stesso.

2.4 Monitoraggio dell'attività utente

Non è presente un monitoraggio persistente delle attività degli utenti ma, per evitare che le sessioni degli utenti rimangano aperte esponendo i dati a potenziali accessi indesiderati, il sistema prevede un timeout di 10 minuti dopo il quale viene mostrato un messaggio che allerta l'utente dell'imminente chiusura della sessione nel caso in cui non venga eseguita nessuna azione.

2.5 Blocco di IP sospetti

Nell'applicazione sono implementate alcune regole per bloccare l'accesso da parte di indirizzi IP che eseguano attività sospette. L'elenco degli IP bloccati è gestibile da pannello amministrativo.

CHAPTER

THREE

ENTITÀ

Di seguito vengono presentate le principali entità di cui è composta l'applicazione:

attachment
Contiene le informazioni di tutti i files caricati nel sistema

attachment_entity
Definisce la relazione tra un file caricato e un'entità

banned_ip
Definisce un elenco di indirizzi ip bloccati a seguito di attività considerate anomale o sospette

dictionary
Dizionari trasversali all'applicazione;

drop_out_row
Informazioni legate al modulo di drop out, rappresenta la singola riga di segnalazione

e_model
Informazioni legate al modello E

e_model_code
Codici diagnosi inseriti in un modello E

ed_salute
Informazioni relative ai progetti di Educazione alla Salute

icf
Contiene la struttura ad albero della tassonomia ICF-CY;

icf_metadata
Associazione dei nodi della tassonomia ICF-CY ai relativi dati descrittivi;

icf_modifier
Modificatori ICF;

internal_message
Contiene i messaggi del sistema di messaggistica interno;

internal_message_module
Definisce per quali moduli il messaggio risulta visibile;

journal
Rappresenta un'azione di modifica su una entità;

journal_row
Dettaglio delle singole modifiche avvenute nell'azione complessiva;

log
Messaggi di log generati dall'applicazione per tracciare processi e anomalie;

mesage
Messagi inviati agli utenti tramite il sistema di notifiche;

metadata
Metadati associati alle entità;

nazione
Elenco delle nazioni;

notice
Notifiche generati da eventi di sistema;

notice_user
Associazione di notifiche agli utenti, data una notifica si può avere la necessità di informare più utenti dell'evento;

notice_user_conf
Sistema di configurazione delle notifiche, permette di stabilire la relazione di associazione automatica tra utenti e notifiche;

pdp_node
Contiene la tassonomia visualizzata per i modelli PDP. La tassonomia è stata separata dalla tassonomia PEI

pdp_node_question
Domande relative ai nodi della tassonomia PDP

pei_node
Tassonomia PEI;

pei_node_question
Domande di riferimento associate ad un nodo della tassonomia PEI;

pei_row
Associazione di un nodo della tassonomia ad un documento PEI;

pei_row_target
Definizione di obiettivi ed attività associati ad una riga del PEI;

property
Impostazioni tecniche dell'applicazione;

role
Ruoli applicativi

role_action
Associazione dei ruoli ad Action e packages;

scheduler
Definisce il piano dei task schedulati e il loro stato;

school_year
Anni scolastici;

somministrazione
Rappresenta l'insieme di prove da somministrare nel modulo "Mondo delle parole progetto letto scrittura" in un intervallo di date per una specifica classe

somministrazione_test
I test inseriti in una somministrare

somministrazione_test_result_class
Le annotazioni relative ad una classe che ha svolto il test di una somministrare

somministrazione_test_result_class_view
Gli utenti che hanno visualizzato le note inseite nelle restituzioni di classe

somministrazione_test_result_user
I risultati di un utente per un test

somministrazione_test_result_user_variable
I risultati di una variabile per un utente in un test;

somministrazione_test_soglia
Le soglie applicate per un test in una specifica somministrazione;

structure
Strutture che nell'applicazione rappresentano le scuole;

structure_class
Le classi presenti in un plesso (struttura);

structure_class_user
Gli utenti presenti in una classe e il loro ruolo;

taxonomy_node
Definizione della tassonomia applicata al modello PEI-PDP;

taxonomy_node_metadata_value
Dati aggiuntivi ai nodi di tassonomia;

taxonomy_node_target
Rappresenta un obiettivo per un nodo della tassonomia;

taxonomy_node_target_activity
Rappresenta un'attività per un obiettivo della tassonomia;

test
Un test caricato nella piattaforma;

test_soglia
I valori di soglia presenti per un test

user
Utenti censiti nell'applicazione;

user_access
Storico degli accessi al sistema da parte di ogni utente;

user_attachment
Allegati legati all'entità utente usati per il portfolio docenti in anno di prova. La gestione dovrebbe passare al sistema centrallizato degli allegati.

user_metadata
Metadati dell'utente, definiscono aspetti estensivi dell'entità;

user_pcto
Contiene i dati del PCTO redatto nel PEI;

user_role
Ruoli dell'utente;

user_structure
Associazione di un utente ad una struttura/scuola;

user_structure_metadata
Metadati legati all'entità user_structure;

user_structure_role
Ruoli rivestiti da un utente in una struttura;

user_transfer_request
Richieste di trasferimento di utenti tra le strutture;

user_user
Associazione di gestione tra utenti. Ad esempio rappresenta i diritti di accesso da parte di un insegnante ai dati di uno studente;

user_year
Dati aggiuntivi applicati ad uno studente;

user_year_document
Documento di uno studente in un anno scolastico;

CHAPTER

FOUR

API

Futura implementa un numero limitato di API. Tali API sono nate per la volontà di consentire a sviluppatori terzi di realizzare moduli esterni in grado di integrarsi con l'applicazione principale. Le API, come ogni altro handpoint del sistema sono sottoposte alla gestione dei ruoli e dei diritti connessi. Questo significa che è possibile configurare ruoli che consentano l'accesso solo ad una parte delle API. Al momento l'unica integrazione presente è con il servizio di valutazione. L'utente associato vede tutti gli handpoint presenti. Di seguito la descrizione degli handpoints.

4.1 Autenticazione

Le API richiedono che le chiamate avvengano da utenti autorizzati. L'autorizzazione avviene inserendo un'api key ottenuto in fase di creazione dell'utente nel seguente header:

X-Apikey: [api_key]

Ovviamente, questa modalità di autenticazione è funzionale alle sole integrazioni server to server. L'api key è da considerare un secret che non deve essere accessibile all'utente o esposto in aplpicazioni SPA o, in generale, nel frontend.

4.2 Accesso dell'utente

URL: {{url}}/api/user/UserAction

Metodo: GET

Richiesta:
token: String

Risposta:
400: token non specificato

404: token non valido o utente non online

200: Ritorna un oggetto contenente le informazioni relative all'utente a cui è associato il token

Questo metodo viene utilizzato per consentire un passaggio trasparente (senza la necessità di eseguire un login separato) dall'applicazione Futura al modulo esterno richiesto dall'utente. Navigando nel sistema l'utente aprirà una pagina conentente un url del tipo:

https://{[}modulo_esterno{]}/?token={[}token]

In questo modo il modulo esterno riceve il token da fornire alla chiamata. Se la chiamata va a buon fine il token è valido, quindi l'applicazione esterna recupera i dati dell'utente: * dati anagrafici * associazione agli istituti * ruoli globali e a livello di istituto

Valutando questi risultati il modulo esterno può decidere se consentire all'utente di proseguire nella navigazione.

4.3 Elenco degli uteni

URL: {{url}}/api/user/UsersAction

Metodo: POST

Richiesta:

```
{
    "first": 0,
    "limit": 10,
    "conditions": {
        "query": "",
        "meccanografico": "",
        "role_id": 1
    }
}
```

first: primo elemento da recuperare.

limit: numero di elementi da recuperare (massimo 100).

conditions: insieme di condizioni/filtri da applicare all'estrazione. Sono attualmente implementati i seguenti filtri:

query: ricerca generica per nome, cognome o email.

meccanografico: estrae solo gli utenti presenti in un certo istituto.

role_id: estra solo gli utenti con un certo ruolo.

Risposta:

400: richiesta non valida

200: Elenco degli utenti che rispettano i criteri, ad esempio:

```
{
    "count": 4,
    "items": [
        {
            "user_id": 18845,
            "email": "insegnante.annodiprova@mbuccio.it",
            "name": "Insegnante",
            "surname": "Anno di prova"
        },
        ....
    ]
}
```