**Spécification technique –api-admin**

**support d’une solution open source d’ENT pour les EPLE**

**de la région Île-de-France**

**Spécification Technique**

**api-admin**

**Auteur** :

**Version** : 1.1

****

**Gestion des changements de version**

*Ce tableau gère les modifications apportées au document au-delà de sa version initiale. Les petites modifications de type erreurs de frappe ou changements de syntaxe ne font pas l’objet d’un suivi. Toute nouvelle version du document ne conserve pas systématiquement les changements apportés lors de la version précédente.*

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Rédacteur** | **Objet de la mise à jour** |
| 1.0 | 30/03/2011 | FCAR | Initialisation du document |
| 1.1 | 26/07/2011 | MMAU | Remaniement |
|  |  |  |  |
|  |  |  |  |

Sommaire

[Objet du document 5](#_Toc299456084)

[1. Fonctionnalités proposées par l’api 5](#_Toc299456085)

[1.1. Objectif 5](#_Toc299456086)

[1.2. Grandes lignes fonctionnelles 5](#_Toc299456087)

[2. Découpage de l’api 6](#_Toc299456088)

[2.1. Découpage technique 6](#_Toc299456089)

[2.2. Découpage fonctionnel des business 6](#_Toc299456090)

[2.2.1. Parties métier communes aux différents modules 6](#_Toc299456091)

[2.2.2. Paramétrage individuel de chaque module 7](#_Toc299456092)

[2.2.3. Gestion des erreurs et des logs 7](#_Toc299456093)

[2.2.4. Suivi de l’alimentation de l’annuaire de l’ENT 8](#_Toc299456094)

[2.2.5. Services tiers en SSO 9](#_Toc299456095)

[2.2.6. Autres 9](#_Toc299456096)

[3. Modèle de données de la base admin 9](#_Toc299456097)

[3.1. Parties communes aux différents modules 10](#_Toc299456098)

[3.1.1. Activation des services 10](#_Toc299456099)

[3.1.1. Logiciels de vie scolaire 11](#_Toc299456100)

[3.1.2. Droits d’accès aux services 11](#_Toc299456101)

[3.1.3. Page publique 12](#_Toc299456102)

[3.2. Paramétrage individuel de chaque module 13](#_Toc299456103)

[3.2.1. Cahier de texte 13](#_Toc299456104)

[3.2.1. Espace de stockage 13](#_Toc299456105)

[3.2.2. Page blanche (web-annuairepb) 14](#_Toc299456106)

[3.2.3. Messagerie 14](#_Toc299456107)

[3.2.4. Serveur ftp 15](#_Toc299456108)

[3.2.5. Logo 15](#_Toc299456109)

[3.2.6. Visionneuses et outils bureautiques 16](#_Toc299456110)

[3.3. Gestion des erreurs et des logs 16](#_Toc299456111)

[3.3.1. Les demandes et leur référentiel 16](#_Toc299456112)

[3.3.2. Journalisation 17](#_Toc299456113)

[3.3.1. Support JIRA 18](#_Toc299456114)

[3.4. Suivi de l’alimentation de l’annuaire de l’ENT 18](#_Toc299456115)

[3.5. Services tiers en SSO 19](#_Toc299456116)

[3.5.1. Ressources numériques 19](#_Toc299456117)

[3.5.1. Orientation 20](#_Toc299456118)

[3.6. Autres 20](#_Toc299456119)

[3.6.1. Xiti 20](#_Toc299456120)

[4. Fonctionnement sous-jacent de l’api 21](#_Toc299456121)

[4.1. Performances et bonne utilisation 21](#_Toc299456122)

[4.2. Configuration de l’api 21](#_Toc299456123)

[4.2.1. Les paramètres de configuration spécifiques 21](#_Toc299456124)

[4.2.2. Dépendances maven spécifiques 21](#_Toc299456125)

[4.3. Gestion des accès à un service 21](#_Toc299456126)

[4.3.1. AccesGestionDto 22](#_Toc299456127)

[4.3.2. sauvegarderBlogDto 22](#_Toc299456128)

[4.3.3. getBlogDtoParEtablissement 24](#_Toc299456129)

[4.4. Gestion des services 26](#_Toc299456130)

# Objet du document

Le but de ce document est de décrire l’api-admin de l’ENT selon ces quatre aspects :

* Les fonctionnalités proposées par l’api
* Le découpage technique
* Le modèle de la base de données
* Les parties complexes de l’api

# Fonctionnalités proposées par l’api

## Objectif

L’api-admin est une api du socle de l’ENT, elle regroupe tous les traitements d’accès et règles de gestion concernant les données de la base d’administration, qui sont principalement les données de paramétrages des services saisies par la console.

L’api-admin permet :

* les traitements d’accès et règles de gestion concernant les données de la base d’administration
* la modification de données principalement par la console d’admin
* la consultation des infos de la base admin par les modules via api-portail

## Grandes lignes fonctionnelles

L’api-admin permet de :

* créer/modifier/consulter les droits d’accès aux différents modules
* consulter la liste des services d’un établissement
* initialiser les services sur un établissement
* activer/désactiver les services facultatifs d’un établissement
* activer/désactiver la page publique d’un établissement
* gérer les services tiers
* paramétrer les services dans la console d’administration

# Découpage de l’api

## Découpage technique

Le découpage en couches business et dao suit les normes de développement définies pour l’ensemble de l’ENT. Le détail de ce fonctionnement est décrit dans le guide de développement général et back-office.

La couche business est directement utilisable par les modules intégrant l’api. L’api-admin a sa base de données propre.

## Découpage fonctionnel des business

### Parties métier communes aux différents modules

*ServicesCategorieBusiness* gère l’activation des modules :

* Activation
* Désactivation
* Consultation des services actifs par établissement
* Consultation de la liste complète des catégories
* Consultation des catégories de services par établissement
* Lien service/vue ria-admin-ihm
* La liste des logiciels vie scolaire
* Le cahier de texte d’un établissement

*AccesGestionServiceBusiness* gère les droits sur les modules :

* Pour la console d’administration, des méthodes de gestion des droits d’accès aux modules et des profils gestionnaires de module
* Pour les modules, des méthodes de consultation de ces droits d’accès

*PagePubliqueBusiness* gère les méthodes liées au paramétrage de la page publique :

* Ajout/Suppression des éditeurs de page de publique
* Ajout/Modification/Suppression de la page publique pour un établissement
* Test pour savoir si un service peut être sur la page publique
* Test pour savoir si une personne est éditeur page publique
* Consultation des paramètres de la page publique

### Paramétrage individuel de chaque module

Chacun de ces business regroupe des méthodes à la fois de paramétrage par la console d’admin, et de consultation de ces paramètres par le module concerné.

Certains modules n’apparaissent pas ici car ils n’ont pas de paramétrage spécifique.

*BureauNumeriqueBusiness* :

* des méthodes pour l’espace de travail, par rapport à l’espace de stockage

*CommunicationMessagerieBusiness* :

* s’occupe des paramètres de la messagerie

*CommunicationBusiness* :

* paramétrage des droits pour le module web-annuaire et le chat qui concernent des données de personnes

*GestionRessourcesBusiness* :

* paramétrage du module gestion de ressources, c'est-à-dire la réservation de matériel

*ServeurFTPAdminBusiness* :

* paramétrage pour les serveurs d’accès distant FTP des établissements

*LogoBusiness*:

* Paramétrage du module web-logos :
  + Ajout/Modification/Suppression d’un logo pour un établissement
  + Consultation de la liste complète des logos d’un établissement
  + Consultation de la liste des logos « visibles » par le module

*SignetPorteurBusiness* :

* les méthodes pour les signets par défaut d’un porteur, c'est-à-dire le module visionneuse et outils bureautiques.
* A noter que ce module possède des tables dans la base admin en prévision d’une future évolution pour rendre le module visionneuse et outils bureautiques modifiable par la console d’admin.

### Gestion des erreurs et des logs

*DemandeBusiness* gère les méthodes de la remontée d’incident par les utilisateurs, pour l’admin et le module web-incident :

* Ajout pour le module web-incident
* Ajout/Modification/Suppression et consultation pour la console d’administration
* Consultation par critères pour la console d’administration
* Consultation des incidents par le journal

*JournalBusiness* :

* Consultation du journal des logs
* Ajout dans le journal

*ReferentielBusiness* s’occupe de la consultation des tables de référentiels pour le module de gestion des demandes :

* Etat des demandes
* Type de demande
* La liste des services

*JiraAdministrationBusiness* gère les méthodes liées au support JIRA qui est accessible depuis la console d’admin :

* Ajout/Consultation des identifiants jira d’un user
* Activation/Désactivation du support sur un établissement
* Test pour savoir si le support jira est activé sur le porteur
* Il ne s’occupe pas des appels en web-service à jira (délégués à l’api-jira).

### Suivi de l’alimentation de l’annuaire de l’ENT

Afin que le traitement d’alimentation automatique prenne en compte les bons lots d’établissements dans les fichiers xml des académies, il faut déclarer dans la console d’administration les établissements à déployer.

Des fichiers de déploiement permettent de déclarer une liste d’établissements en une seule fois.

*EtablissementAdminBusiness* :

* Ajout d’un établissement à déployer
* Test de l’existence d’un établissement

*FichierDeploiementBusiness* :

* Ajouter/Modifier/Consulter les infos des fichiers de déploiement.

Il ne gère pas le contenu des fichiers de déploiement mais les infos annexes :

* + Nom
  + Académie
  + Etablissement
  + Chargé ou non dans l’ent

### Services tiers en SSO

*OrientationBusiness* traite les méthodes liées à la page orientation :

* Ajout/Modification/Suppression et consultation de liens de la page orientation
* Ajout/Modification/Suppression et consultation de messages de la page orientation

*RessourceBusiness :*

* La consultation de la liste des ressources service tiers, vie scolaire et BCDI
* La consultation de la liste des ressources accessibles service tiers, vie scolaire et BCDI
* La gestion des abonnements pour service tiers, vie scolaire et BCDI

### Autres

*XitiBusiness* regroupe les méthodes pour les statistiques centralisées par l’outil XITI de la région.

# Modèle de données de la base admin

Pour des facilités de lecture le modèle a été séparé par domaine fonctionnel.

Le principal lien entre ces domaines est la table ETABLISSEMENT, car la plupart des données se consultent et se gèrent par établissement :

## Parties communes aux différents modules

### Activation des services

Ces tables regroupent les infos concernant les services et les catégories, et leurs relations entre eux et avec un établissement et les panels de ria-admin-ihm pour les vues de l’ihm.

La table rel\_etab\_service a une relation avec etablissement.

Le business pour ces tables est *ServicesCategorieBusiness.*

### Logiciels de vie scolaire

Ces tables sont les données pour les logiciels de vie scolaire.

La table rel\_etab\_vie\_sco a une relation avec etablissement.

Le business qui se base sur ces tables est *ServicesCategorieBusiness*.

### Droits d’accès aux services

Ce sont les tables qui gèrent les accès aux services par profil, et les droits de gestion par profil pour un service et un établissement.

La table acces\_gestion\_service a une relation avec etablissement.

Le business associé est *AccesGestionServiceBusiness.*

### Page publique

Ces tables sont liées à la gestion des pages publiques.

Les tables page\_pubique et gestion\_page\_publique ont une relation avec etablissement.

Leur business est *PagePubliqueBusiness*.

## Paramétrage individuel de chaque module

### Cahier de texte

Ces tables sont les tables pour le module cahier de texte.

La table rel\_etab\_cdt a une relation avec etablissement.

Le business est *ServicesCategorieBusiness*.

### Espace de stockage



Cette table contient les données de paramétrage des espaces de travail.

Son business est *BureauNumeriqueBusiness.*

### Page blanche (web-annuairepb)



Ces deux tables concernent les pages blanches, en particulier les droits de consultation par profil aux pages blanches.

La table page\_blanche a une relation avec etablissement.

Le business est *CommunicationBusiness*.

### Messagerie



Ces deux tables sont les tables pour les droits d’accès à la messagerie. Le business associé est *CommunicationMessagerieBusiness*.

### Serveur ftp



Cette table sert pour le module ftp, il permet de connaitre l’url du serveur distant d’un établissement.

Le business de cette table est *ServeurFTPAdminBusiness*.

La table a une relation avec etablissement.

### Logo

Ce sont les tables de logo et leur relation avec un établissement pour le module logo.

Il n’y a pas d’autres tables spécifiques et d’autre base pour le module logo.

La table rel\_etablissement\_logo a une relation avec etablissement.

Le business est *LogoBusiness*.

### Visionneuses et outils bureautiques

Ces tables sont l’ensemble des tables spécifiques pour le module « Visionneuse et outils bureautiques », il n’y a pas d’autre base de données pour ce module.

Le business est *SignetPorteurBusiness.*

## Gestion des erreurs et des logs

### Les demandes et leur référentiel

Ces tables s’occupent des demandes et de leur référentiel.

Le business *DemandeBusiness* s’occupe de la table demande et demandeur.

A noter que la table « demande » a deux relations supplémentaires :

* avec la table service
* avec la table etablissement

Les autres tables sont des tables de référentiel, et la consultation est effectuée par le business *ReferentielBusiness*.



### Journalisation

Ce sont les tables pour la journalisation.

Le business associé est *JournalBusiness*.

### Support JIRA

Ces tables servent pour le support JIRA, elles contiennent les infos des utilisateurs JIRA, et l’activation du support JIRA pour un établissement et un porteur.

Ces tables sont utiles pour le business *JiraAdministrationBusiness*.



La table rel\_etab\_jira a une relation avec la table etablissement.

## Suivi de l’alimentation de l’annuaire de l’ENT

Cette table regroupe les infos des fichiers de déploiement.

Le business de cette table est *FichierDeploiementBusiness*.

## Services tiers en SSO

### Ressources numériques

Ces tables sont les tables des ressources numériques, elles gèrent les services tiers, bcdi, et vie scolaire et leurs relations avec un établissement et un porteur.

La table « rel\_ressource\_etablissement » a une relation supplémentaire avec la table etablissement.

Le business pour ces tables est *RessourceBusiness.*

### Orientation

Ces deux tables sont pour la page orientation, les liens et les messages sur cette page.

Les deux tables ont une relation avec etablissement.

Leur business est *OrientationBusiness*.

## Autres

### Xiti

Ces tables s’occupent de xiti, leur business est *XitiBusiness*.



# Fonctionnement sous-jacent de l’api

## Performances et bonne utilisation

TODO

## Configuration de l’api

### Les paramètres de configuration spécifiques

Il n’y a pas de configuration spécifique pour cette api. Les seules propriétés dans le *config.properties* sont pour la définition de la base de données. Toutes les configurations (config, spring, dozer etc…) suivent les normes de développement ENT définies dans le guide de développement.

### Dépendances maven spécifiques

L’artifact correspondant à cette api est le suivant :

*<groupId>org.lilie.socle</groupId>*

*<artifactId>api-admin</artifactId>*

Les dépendances de l’api ne sont pas spécifiques, ce sont simplement les librairies concernant les couches back-office des applications.

## Gestion des accès à un service

D’un point de vue fonctionnel, c’est l’api-admin qui centralise les traitements permettant de :

* restreindre l’accès aux modules en fonction du profil de l’utilisateur



* définir les profils et/ou fonctions possédant le rôle « gestionnaires de service »



Ces paramètrages sont accessibles dans la console d’administration.

Dans notre exemple, nous traiterons le cas du blog mais le comportement est globalement le même pour les autres services.

Dans le business *AccesGestionServiceBusiness*, on retrouve deux méthodes :

* Pour la sauvegarde des droits

**void** sauvegarderBlogDto(AccesGestionDto blogDto)

* Pour la récupération de ces droits

AccesGestionDto getBlogDtoParEtablissement

L’objet utilisé pour le transfert de ces données est AccesGestionDto. Cet objet est commun à tous les services.

### AccesGestionDto

Ce DTO contient trois types de droits :

* droit d’accès à un service
* droit de gestionnaire de service
* droit d’export des données d’un service

Chacun de ces droits est décliné par profil (eleve, enseignant, autre, personnel et parent).

Ces informations se présentent par un boolean par type de droit et par profil.

De plus le personnel peut être géré plus finement, par fonction.

### sauvegarderBlogDto

Cette méthode enregistre les infos dans la table rel\_acces\_gestion\_service

/\*\*

\* Methode permettant d'enregistrer les données blog dans la base.

\* **@param** blogDto le dto blog

\* **@throws** ServiceTechniqueException :

\* **@throws** ServiceFonctionnelleException :

\*/

**public** **void** sauvegarderBlogDto(AccesGestionDto blogDto)

**throws** ServiceTechniqueException, ServiceFonctionnelleException {

*LOGGER*.debug("entree dans la methode");

**try** {

sauvegarder(adapter.map(blogDto, AccesGestionService.**class**), Constantes.*TYPE\_SERVICE\_BLOG*);

} **catch** (ServiceFonctionnelleException dfe) {

**throw** dfe;

} **catch** (ServiceTechniqueException dte) {

**throw** dte;

} **catch** (Exception e) {

**throw** **new** ServiceTechniqueException(e.getMessage(), e);

}

}

}

}

Chaque service a une méthode particulière qui appelle une méthode générique sauvegarder avec en paramètre le service concerné.

La méthode générique :

/\*\*

\* Sauvegarde du pojo AccesGestionService

\* **@param** accesGestionServiceCommun AccesGestionService

\* **@param** typeStr type de service

\* **@throws** ServiceTechniqueException leve une exception technique

\* **@throws** ServiceFonctionnelleException leve une exception fonctionnelle

\*/

**private** **void** sauvegarder(AccesGestionService accesGestionServiceCommun, String typeStr)

**throws** ServiceTechniqueException, ServiceFonctionnelleException {

*LOGGER*.debug("entree dans la methode");

**try** {

AccesGestionService accesGestionService = accesGestionServiceDao

.getAccesGestionServiceParEtablissement(accesGestionServiceCommun.getEtablissementId(), typeStr);

**if** (accesGestionService == **null**) {

accesGestionServiceDao.ajouterAccesGestionServiceParEtablissement(

accesGestionServiceCommun.getEtablissementId(), typeStr);

} **else** {

accesGestionServiceCommun.setTypeService(typeStr);

accesGestionServiceDao.majAccesGestionService(accesGestionServiceCommun);

}

} **catch** (DaoFonctionnelleException dfe) {

*LOGGER*.info(dfe.getMessage());

**throw** **new** ServiceFonctionnelleException(dfe.getErrorCode(), dfe.getMessage());

} **catch** (DaoTechniqueException dte) {

*LOGGER*.error(dte.getMessage());

**throw** **new** ServiceTechniqueException(dte.getMessage(), dte);

} **catch** (Exception e) {

*LOGGER*.error(e.getMessage(), e);

**throw** **new** ServiceTechniqueException(e.getMessage(), e);

}

}

Les services en paramètres sont des constantes définies dans

« /api-admin/src/main/java/org/lilie/socle/api/admin/utils/Constantes.java »

**public** **static** **final** String *TYPE\_SERVICE\_BLOG* = "blogs";

### getBlogDtoParEtablissement

Cette méthode récupère les infos utilisées pour savoir si le service doit être affiché dans le menu du portail et dans le service pour savoir si le profil connecté est gestionnaire.

/\*\*

\* Methode recuperant les données blog par rapport à l'id de l'etablissement

\* **@param** educId id de l'etablissement

\* **@return** blogDto le dto blog de l'etablissement

\* **@throws** ServiceTechniqueException :

\* **@throws** ServiceFonctionnelleException :

\*/

**public** AccesGestionDto getBlogDtoParEtablissement(**long** educId)

**throws** ServiceTechniqueException, ServiceFonctionnelleException {

*LOGGER*.debug("entree dans la methode");

**try** {

**return** adapter.map(getAccesGestionServiceParEtablissement(educId, Constantes.*TYPE\_SERVICE\_BLOG*),

AccesGestionDto.**class**);

} **catch** (ServiceFonctionnelleException dfe) {

**throw** dfe;

} **catch** (ServiceTechniqueException dte) {

**throw** dte;

} **catch** (Exception e) {

**throw** **new** ServiceTechniqueException(e.getMessage(), e);

Chaque service a une méthode particulière qui appelle une méthode générique getAccesGestionServiceParEtablissement avec en paramètre le service concerné.

/\*\*

\* Recupere le pojo AccesGestionService par etablissement et par type

\* **@param** etablissementId identifiant de l'etablissement

\* **@param** typeStr type de service

\* **@return** AccesGestionService

\* **@throws** ServiceTechniqueException leve une exception technique

\* **@throws** ServiceFonctionnelleException leve une exception fonctionnelle

\*/

**private** AccesGestionService getAccesGestionServiceParEtablissement(Long etablissementId, String typeStr)

**throws** ServiceTechniqueException, ServiceFonctionnelleException {

**try** {

AccesGestionService accesGestionServiceDto = accesGestionServiceDao

.getAccesGestionServiceParEtablissement(etablissementId, typeStr);

**if** (accesGestionServiceDto == **null**) {

accesGestionServiceDto = **new** AccesGestionService();

accesGestionServiceDto.setEtablissementId(etablissementId);

accesGestionServiceDto.setTypeService(typeStr);

sauvegarder(accesGestionServiceDto, typeStr);

accesGestionServiceDto = accesGestionServiceDao

.getAccesGestionServiceParEtablissement(etablissementId, typeStr);

}

**return** accesGestionServiceDto;

} **catch** (DaoFonctionnelleException dfe) {

*LOGGER*.info(dfe.getMessage());

**throw** **new** ServiceFonctionnelleException(dfe.getErrorCode(), dfe.getMessage());

} **catch** (DaoTechniqueException dte) {

*LOGGER*.error(dte.getMessage());

**throw** **new** ServiceTechniqueException(dte.getMessage(), dte);

} **catch** (Exception e) {

*LOGGER*.error(e.getMessage(), e);

**throw** **new** ServiceTechniqueException(e.getMessage(), e);

}

}

## Gestion des services

Comme vu dans la console d’administration, l’arbre des services d’un établissement est dynamique.



La relation entre la vue du service et le bouton de chaque service est faite grâce à l’api-admin et la méthode getClasse.

/\*\*

\* Retourne la classe d'un panel

\* **@return** la classe correspondant au panel\_id

\* **@param** panelIdInt panel id

\* **@throws** ServiceTechniqueException : exception technique de la couche metier

\* **@throws** ServiceFonctionnelleException : exception fonctionnelle de la couche metier

\*/

**public** String getClasse(**int** panelIdInt)

**throws** ServiceTechniqueException, ServiceFonctionnelleException {

**try** {

**return** servicesCategorieDao.getClasse(panelIdInt);

} **catch** (DaoFonctionnelleException dfe) {

*LOGGER*.info(dfe.getMessage());

**throw** **new** ServiceFonctionnelleException(dfe.getErrorCode(), dfe.getMessage());

} **catch** (DaoTechniqueException dte) {

*LOGGER*.error(dte.getMessage());

**throw** **new** ServiceTechniqueException(dte.getMessage(), dte);

} **catch** (Exception e) {

*LOGGER*.error(e.getMessage(), e);

**throw** **new** ServiceTechniqueException(e.getMessage(), e);

}

}

Cette méthode utilise la table « panel »  qui associe à un panel une classe.

Par exemple pour blog :



Extrait de ria-admin-ihm pour exemple :

/\*\*

\* Execute commande

\* **@param** notification notification

\*/

**public** **void** execute(INotification notification) {

Vue2Mediator vue2Mediator = (Vue2Mediator) getFacade().retrieveMediator(Vue2Mediator.**class**.getName());

WorkbenchMediator workbenchMediator = (WorkbenchMediator) getFacade()

.retrieveMediator(WorkbenchMediator.**class**.getName());

Vue2View view = getClasse(notification.getBody().toString());

Donc grâce à l’id du panel on retrouve la vue recherchée, en passant par l’api-admin et la méthode *getClasse*.