

Brancher AnyBlok à 14 ans d'historique métier

Une histoire terrifiante de PHP, MySQL et MsSQL

Jean-Sébastien SUZANNE et Hugo QUEZADA



2 novembre 2019

Qui sommes nous ?


Sensee.

**lentilles
moinscheres.com**

Sébastien Suzanne

- Répond aussi au nom de PAPABLOK
-  @jssuzanne
-  js.suzanne@sensee.com

Hugo Quezada

- Dis le petit Basque du Chili
-  h.quezada@sensee.com

Existant

14 ans de code...

Existant

14 ans de code...

- Code legacy en PHP5 (pas de troll SVP)
- Pas de framework, beaucoup de code, très peu d'objet

Existant

14 ans de code...

- Code legacy en PHP5 (pas de troll SVP)
- Pas de framework, beaucoup de code, très peu d'objet
- Pas de tests
- Pas de CI

Existant

14 ans de code...

- Code legacy en PHP5 (pas de troll SVP)
- Pas de framework, beaucoup de code, très peu d'objet
- Pas de tests
- Pas de CI
- Pas d'ORM

Existant

Une BdD un peu complexe...

- Schémas de base de données multiples

Existant

Une BdD un peu complexe...

- Schémas de base de données multiples
- Écosystème avec plusieurs SGBD (MySQL, MsSQL) souvent sans API

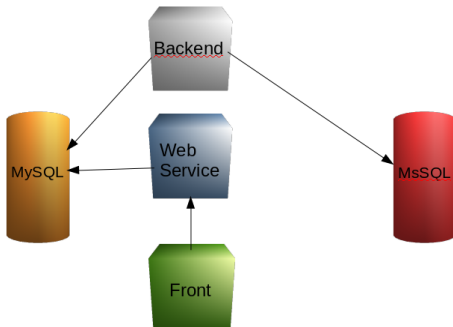


Figure: Schéma simplifié

Vision finale

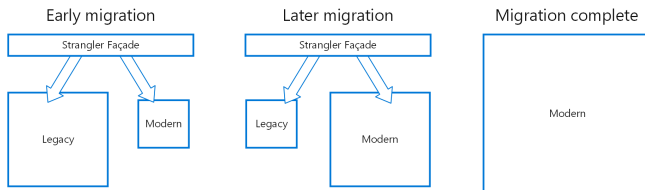
L'objectif

- Un code python propre et testé
- Une API simple et uniforme, utilisé dans tous nos projets
- Une application plus proche des standards actuels
- Un projet plus séduisant et plus attrayant pour des potentiels futurs développeurs

Vision finale

La stratégie

- Modèle d'étranglement (Strangler pattern)



- Mapping des tables avec un nommage plus clair
- Développement piloté par les tests (TDD)

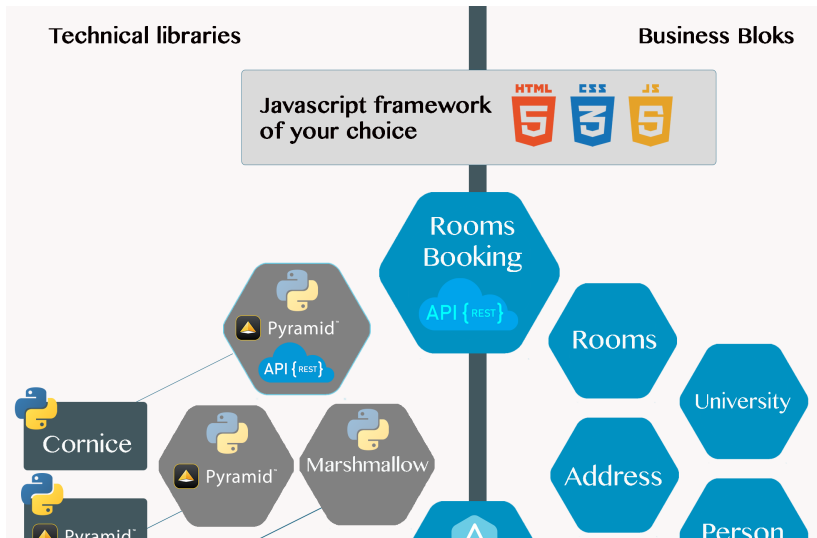
Pourquoi avoir choisi AnyBlok ?

AnyBlok: Présentation

- Python 3.6 et plus
- MPL2
- PyPi
- Modulaire
- Dépendances fiables (SQLAlchemy, Alembic, ...)
- Possibilité de reprendre une base existante.

Pourquoi avoir choisi AnyBlok ?

AnyBlok: Structure



Pourquoi avoir choisi AnyBlok ?

AnyBlok: Écrire des tests pour notre Model

```
import pytest

@pytest.mark.usefixtures('rollback_registry')
class TestProduct:

    def test_create_new_product(self, rollback_registry):
        rollback_registry.LMC.Product.insert(
            label="A product label",
            code="A product code",
        )
        assert rollback_registry.Product.query().count() == 1
        assert (
            rollback_registry.Product.query().one().code ==
            "A product code"
        )
```

Pourquoi avoir choisi AnyBlok ?

AnyBlok: Définir le Model

```
from anyblok import Declarations
from anyblok.column import String, Integer
```

```
Declarations.@register(Declarations.Model.LMC)
class Product:
```

```
    id = Integer(primary_key=True)
    code = String(nullable=False, index=True)
    label = String(nullable=False)
```

Pourquoi avoir choisi AnyBlok ?

AnyBlok: Définir un Model sur une table existante

```
from anyblok import Declarations
from anyblok.column import String, Integer

Declarations.@register(Declarations.Model.LMC, tablename="ProdDef")
class Product:

    id = Integer(primary_key=True, db_column_name="IDD")
    code = String(nullable=False, index=True, db_column_name="Attr1")
    label = String(nullable=False, db_column_name="Display")
```

Pourquoi avoir choisi AnyBlok ?

AnyBlok: Écrire des tests pour notre L'API

```
import pytest

@pytest.mark.usefixtures('rollback_registry')
class TestApiProduct:

    def test_addresses_post(self, rollback_registry, webserver):
        response = webserver.post_json('/api/v1/lmc/products',
                                       [{'code': 'C1', 'label': 'My product'}])
        assert response.status_code == 200
        assert response.json_body[0].get('code') == 'C1'
        assert rollback_registry.LMC.Product.query().filter_by(code='C1').one()

    def test_addresses_get(self, rollback_registry, webserver):
        """Address GET /api/v1/addresses"""
        rollback_registry.LMC.Product.insert(
            code="C1", label="My product")
        response = webserver.get('/api/v1/lmc/products')
        assert response.status_code == 200
        assert len(response.json_body) == 1
        assert response.json_body[0].get('code') == 'C1'
```


Pourquoi avoir choisi AnyBlok ?

AnyBlok: Écrire une API rest

```
from anyblok_pyramid_rest_api.crud_resource import (
    CrudResource, resource)

@resource(
    collection_path='/api/v1/lmc/products',
    path='/api/v1/lmc/products/{id}',
    installed_blok=current_blok()
)
class LensProductResource(CrudResource):
    model = "Model.LMC.Product"
```

Evolutions et intégrations dans AnyBlok

Compatibilité avec MySQL

Evolutions et intégrations dans AnyBlok

Compatibilité avec MySQL

- Deux semaines de travail
- Beaucoup de recherches et d'incompréhensions

Evolutions et intégrations dans AnyBlok

Compatibilité avec MySQL : Les tests unitaires

- Mise à jour de la configuration Travis-CI
- Activer le mode transactionnel de InnoDB
- Les commits implicites

Evolutions et intégrations dans AnyBlok

Compatibilité avec MySQL : Limitations

- Pas de Python 3.5
- Pas de CheckConstraint et autre contrainte exclusive à PostgreSQL
- Datetime naïves
- Pas de chiffrement sur les colonnes de type UUID
- Pas de véritable Booléen

Evolutions et intégrations dans AnyBlok

Compatibilité avec MariaDB : Limitations

- Pas de Python 3.5
- Pas de CheckConstraint et autre contrainte exclusive à PostgreSQL
- Datetime naïves
- Pas de chiffrement sur les colonnes UUID
- Pas de véritable Booléen

Evolutions et intégrations dans AnyBlok

Compatibilité avec MariaDB : Limitations

- Pas de Python 3.5
- Pas de CheckConstraint et autre contrainte exclusive à PostgreSQL
- Datetime naïves
- Pas de chiffrement sur les colonnes UUID
- Pas de véritable Booléen
- Taille des clés primaires plus petite
- Pas de colonnes JSON

Évolutions et intégrations dans AnyBlok

Compatibilité avec MsSQL : Limitations

- Pas de Python 3.5
- Pas de CheckConstraint et autre contrainte exclusive à PostgreSQL

Évolutions et intégrations dans AnyBlok

Compatibilité avec MsSQL : Limitations

- Pas de Python 3.5
- Pas de CheckConstraint et autre contrainte exclusive à PostgreSQL
- Lent... Très très lent...

Évolutions et intégrations dans AnyBlok

Définition de schémas

- Programmatique ou par configuration
- Poser sur un modèle ou un namespace
- Ajout de suffixes ou préfixes pour les tests

Évolutions et intégrations dans AnyBlok

Définition de schémas

- Programmatique ou par configuration
- Poser sur un modèle ou un namespace
- Ajout de suffixes ou préfixes pour les tests

Problématiques

- génération de foreign key
- migration

Évolutions et intégrations dans AnyBlok

Ajouter un schema a un Model

```
from anyblok import Declarations
from anyblok.column import String, Integer

Declarations.@register(Declarations.Model.LMC, tablename="ProdDef")
class Product:
    __db_schema__ = "Lens"

    id = Integer(primary_key=True, db_column_name="IDD")
    code = String(nullable=False, index=True, db_column_name="Attr1")
    label = String(nullable=False, db_column_name="Display")
```

exemple configuration

Interface Graphique

FuretUI

AnyBlok

Documentation

- <https://anyblok.gitbooks.io/anyblok-book/content/en/doc.anyblok.org>
- [doc.anyblok.org](https://anyblok.org)
- <https://github.com/AnyBlok>
- <https://gitter.im/AnyBlok/community>

Fin

Des questions ?
Des remarques ?