Plateforme pour lutter contre les troubles du langage

L'art de bien parler...
C'est l'art de bien pratiquer

Rechercher

SOMMAIRE

Contexte

Créer une plateforme pour lutter contre les troubles du langage chez l'enfant.
Les parents se sentent impuissant face aux troubles dont souffre leurs enfants. Il est urgent de les aider. Cette plateforme va permettre aux parents de pouvoir aider son enfant a améliorer sa communication grâce a un parcours ludique personnalisé.

Les **parents** comme les **patients** ont besoin :

- De pratiquer en dehors des sessions.
- Se documenter sur la pathologie
- Aider son enfant à améliorer son langage au quotidien.

Les **orthophonistes** ont besoin

- D'un outil pour réaliser des compte-rendus, des fiches patients.
- D'organiser des sessions.

Ludic Language

Persona et Trello

Persona

Identifier les profiles des utilisateurs

- Création des personas
  - Patient / Parents
  - Speech Therapist

Impact Mapping et Users Stories

Trello

Fichier Backlog avec les users stories

Ludic Language

Marie

39 ans

Orthophoniste

Paris

"L'art de bien parler c'est l'art de bien pratiqué"

Bio

Marie est passionnée de lecture et de langage. Elle savait très jeune qu'elle voulait en faire son métier. Elle a donc fait des études afin de devenir orthophoniste. Elle a commencé à exercer à l'hôpital. Elle aidait les victimes d'AVC à réapprendre à parler. Puis, elle a décidé de s'orienter vers les enfants. Elle a pris conscience assez tôt des retard de langages chez les jeunes enfants du notamment aux écrans. Elle a donc repris ses études et s'est spécialisé dans le

PERSONNALITE

Extravert — Introvert

Analyste — Creative

Thinking — Feeling

Judging — Perceiving

TECHNOLOGIE

IT and Internet

Software

Mobile Aps

Social Networks

BRANDS

Système suivi des patients Langage Ludique

En tant qu'orthophoniste, Marie a besoin d'un système efficace qui effectue un suivi approfondis.

Besoins

Marie a besoin de réaliser un bilan afin programmer des exercices en fonction du patient. Elle doit suivre les évolutions de son patient et faire des compte rendus pour que les parents et les patients. Elle a besoin d'un système qui planifie les rendez-vous en visio ou sur place.

Plateforme

Proposer une application qui optimise le suivi des patients à distance. Un système que les patients et les parents puissent utiliser assez facilement. Une structure qui permet à chaque orthophoniste de préparer chaque parcours en fonction du patient.

Expertise du domaine

Le système doit proposer divers exercices et permettre aux orthophonistes de pouvoir proposer d'autres exercices en contactant l'administrateur. Les orthophonistes, une fois choisi, pourront proposer un parcours éducatif sur-mesure avec l'aide du profil du patient.

Clarence

9 ans

Ecole élémentaire

Clamart

"Lire sans hésiter sur un mot à cause du bégaiement"

Bio

Clarence est un garçon curieux mais pas très bavard à cause de son bégaiement. Son bégaiement s'accentue lorsqu'il est en classe. Il ne participe jamais en classe du fait de son handicap. Son bégaiement est assez modéré lorsqu'il se trouve dans un cadre familier mais s'accentue lorsqu'il se trouve dans une situation stressante. Il est très doué en jeu vidéo mais aussi en dessin. Il excelle dans les domaines dont la rhétorique est secondaire voire inexistante.

PERSONNALITE

Extravert — Introvert

Analyste — Creative

Thinking — Feeling

Judging — Perceiving

TECHNOLOGIE

IT and Internet

Software

Mobile Aps

Social Networks

Système suivi des patients Langage Ludique

Clarence a besoin d'un système dans lequel il peut s'exercer à parler sans stresser.

Besoins

Il a besoin de pratiquer des exercices à voix haute. Il aimerait s'entendre parler afin de pouvoir améliorer sa diction. Il désire choisir son orthophoniste même s'il doit changer après une séance. Il doit se sentir à l'aise avec la personne qui va s'occuper de lui.

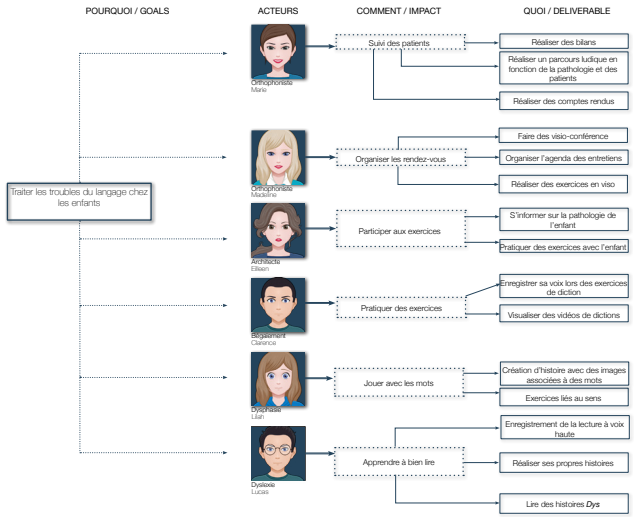
Plateforme

Proposer une application qui permette aux enfants bègues de pouvoir répéter des phrases, des mots et être enregistré. Cette application doit proposer des exercices à plusieurs niveaux mais pas forcément lié à l'âge du patient.

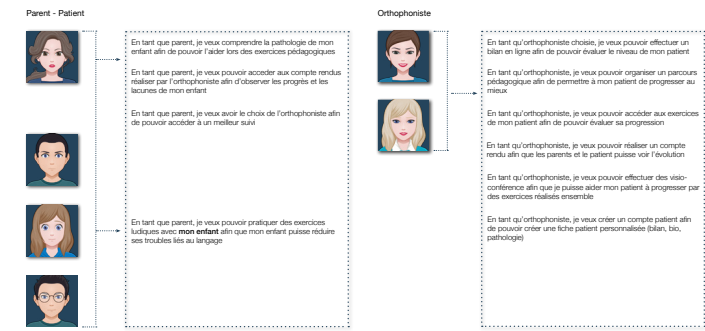
Expertise du domaine

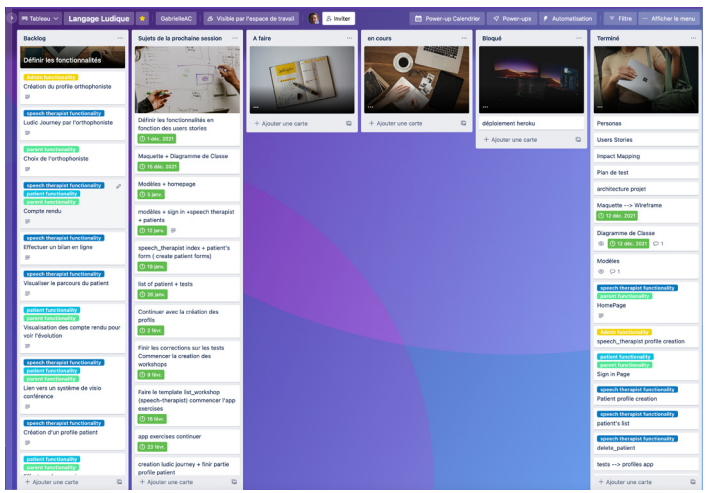
Le système doit proposer des exercices pratiques avec des enregistrements vocaux. Des vidéos qui mettent en scène des exercices de dictions.

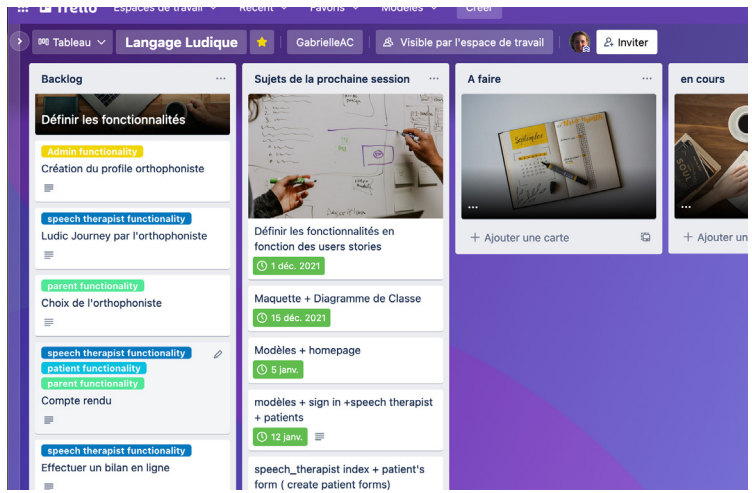
Impact Mapping



Users Stories





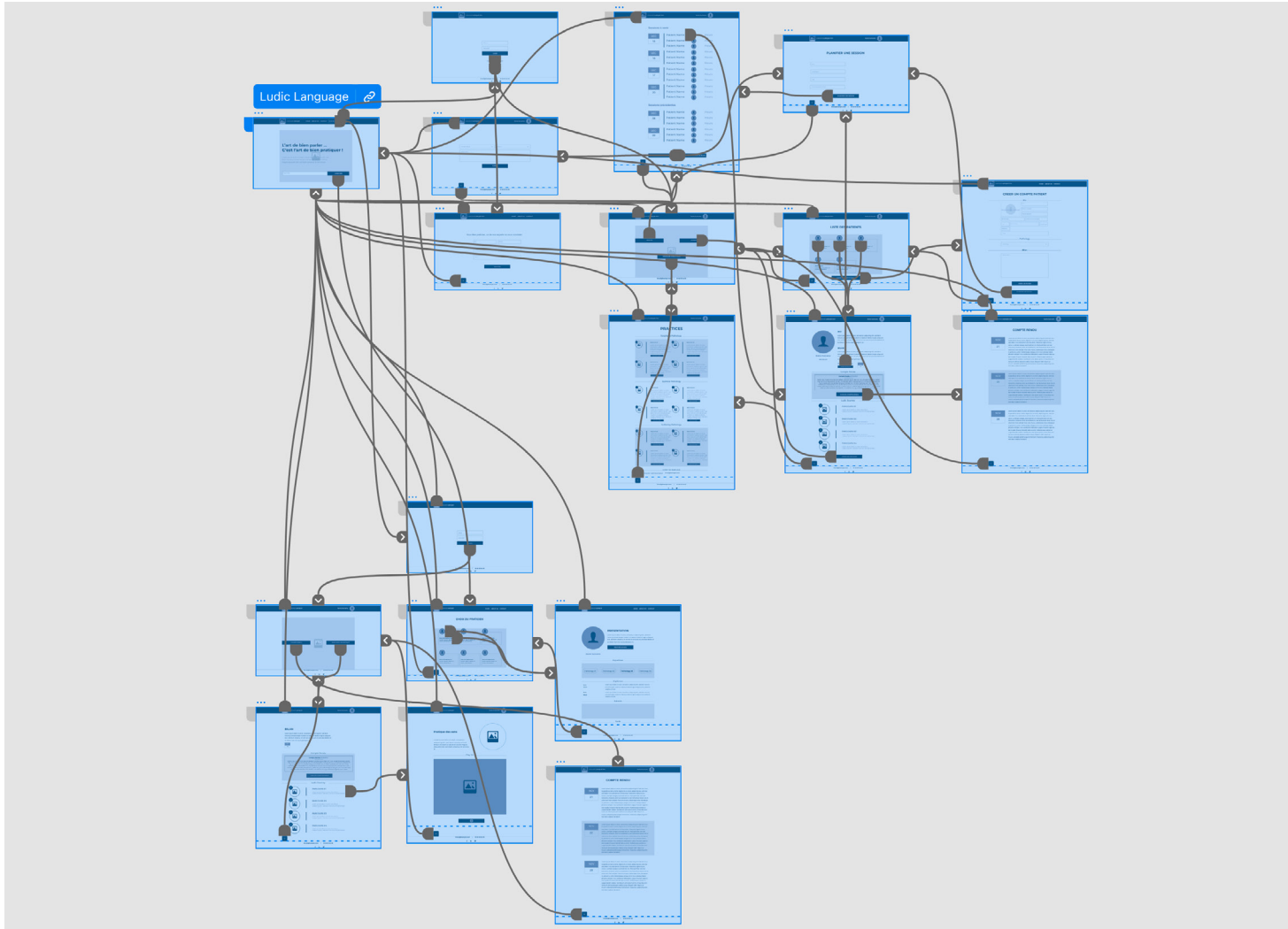


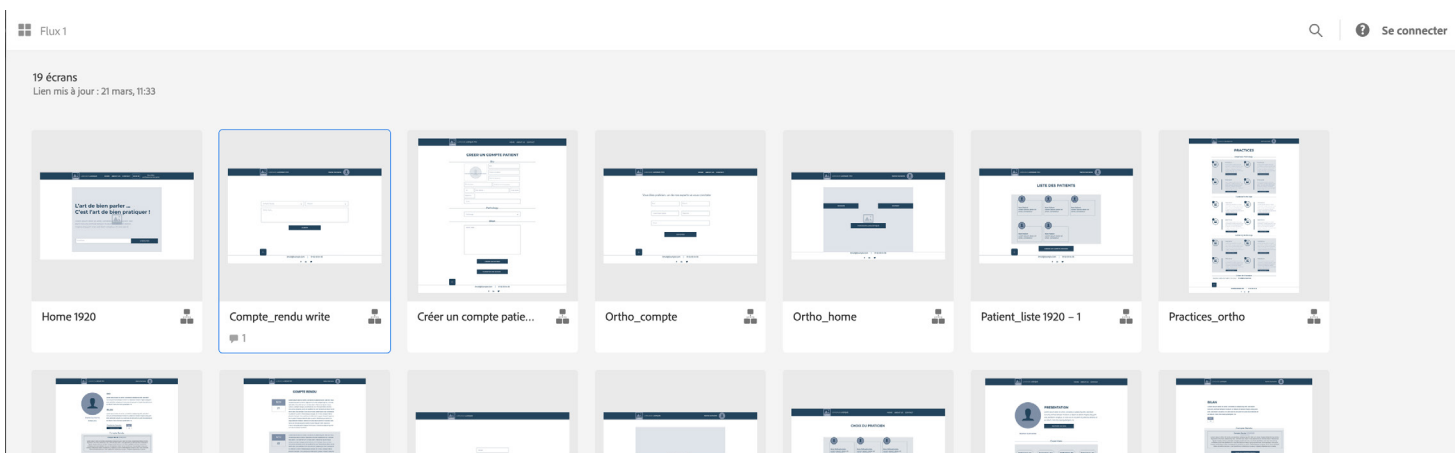
Maquette & Wireframe

XD Adobe

Réalisation d'une maquette sur papier pour poser des idées.

Réalisation de maquette --> Wireframe sur le XD Adobe





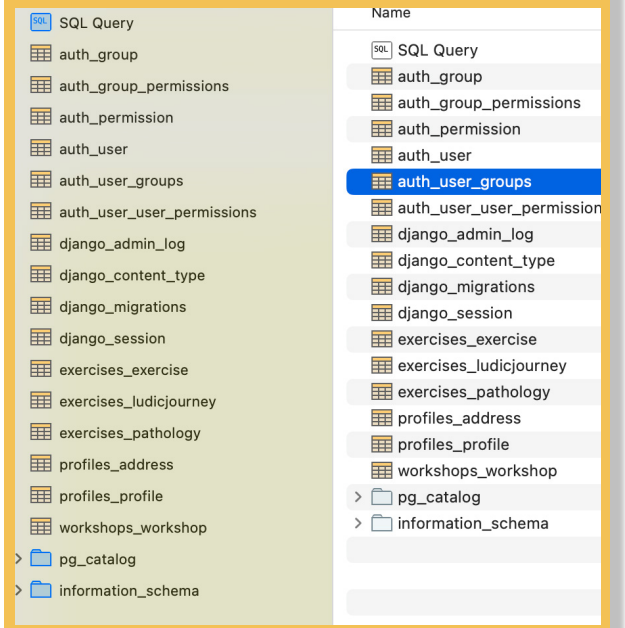
MPD

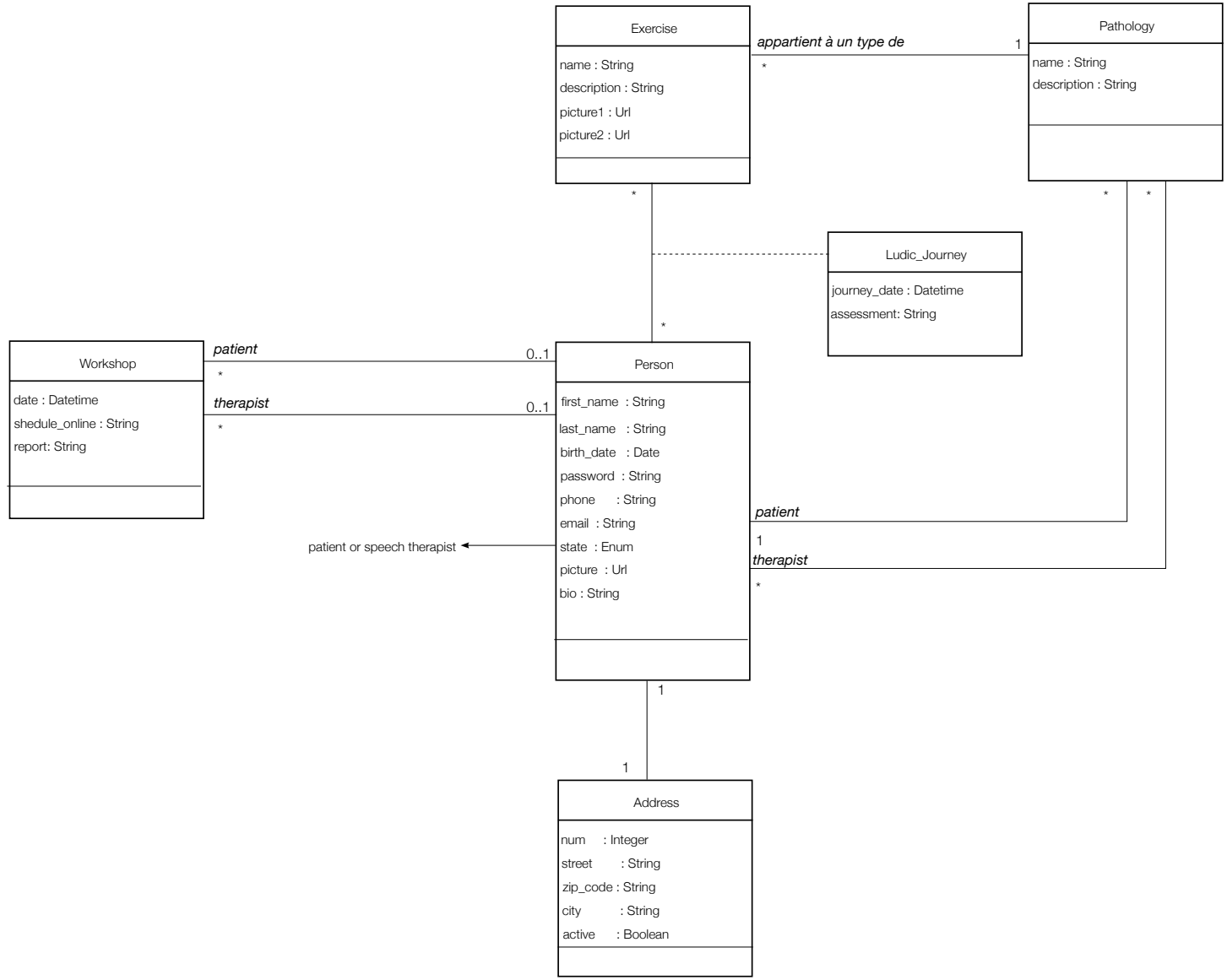
Modèle Physique de Données

Réalisation du MPD sur Indesign

Postico

Visualisation des tables





Ludic Language

Architecture et plan de test

Architecture

Création de l'architecture du projet (Xtreme Programming)

Plan de test

Ludic Language

Projet [P13] Les troubles du langage chez l'enfant -> Architecture

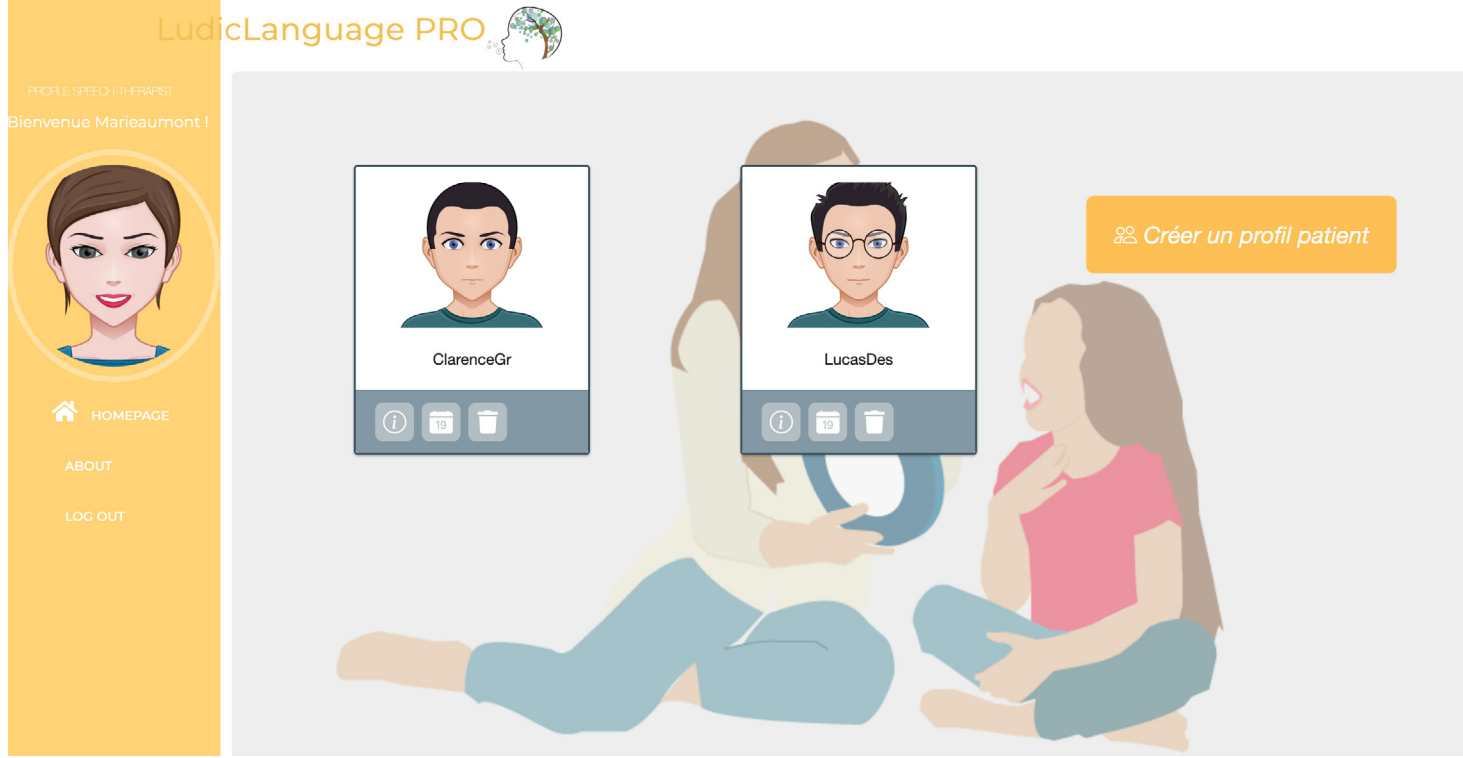
```
language_project/
│
├─ludic_language/
│       ├─ __init__.py
│       ├─ asgi.py
│       ├─ urls.py
│       ├─ wsgi.py
│       ├─ settings.py
│       ├─ base/
│       │    ├─__init__.py
│       │    ├─admin.py
│       │    │
│       │    ├─views.py
│       │    ├─ templates/
│       │    │    ├─index.html
│       │    │    ├─legal_notice.html
│       │    │    └─aboutus.html
│       │    │
│       │    └─ statics/
│       │         ├─css/
│       │         │   └─styles.css
│       │         ├─js/
│       │         │   └─script.js
│       │         └─assets/
│       │              ├─favicon.ico
│       │              └─img/
│       ├─ profiles/
│       │    ├─__init__.py
│       │    ├─admin.py
│       │    │
│       │    ├─views.py
│       │    ├─models.py
│       │    ├─forms.py
│       │    ├─ templates/
│       │    │    ├─login.html
│       │    │    ├─index_patient.html
│       │    │    ├─index_speech.html
│       │    │    ├─patient_list.html
│       │    │    ├─speech_list.html
│       │    │    ├─form_patient.html
│       │    │    └─patient_confirm_delete.html
│       │    ├─ tests/
│       │    │    ├─test_views.py
│       │    │    ├─test_forms.py
│       │    │    ├─test_models.py
│       │    │    └─tests_functional.py
│       │    │
│       │    │
│       │    ├─ statics/
│       │    │    ├─ profiles/
│       │    │    │    ├─css/
│       │    │    │    │   └─styles.css
│       │    │    │    ├─js/
│       │    │    │    │   └─script.js
│       │    │    │    └─assets/
│       │    │    │         └─img/
```

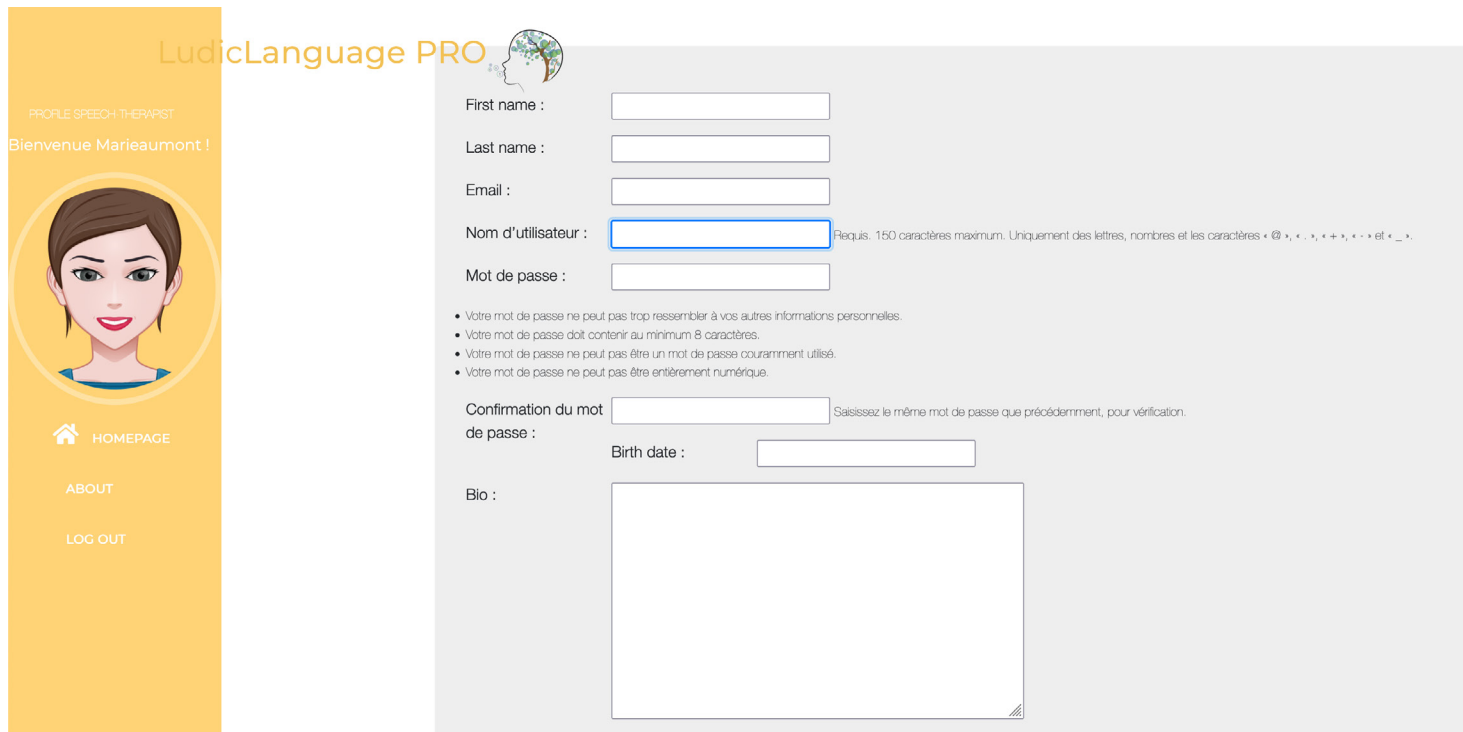
DJANGO

App Profiles

Création du formulaire de création de patient

- Templates





App Profiles

Création du formulaire de création de patient

- Template
- foms.py
- views.py

enctype=»multipart/form-data» --> upload de données

Récupère l'identifiant de l'utilisateur connecté et le passe en argument dans le formulaire

```
{% extends 'index_speech.html' %}
{% block content%}
<div class=»content-container»>
 <div class=»content-container»>
        <section class=»jumbotron»>
          <form method=»POST» enctype=»multipart/form-data»>
                {% csrf_token %}
                {{ form.as_p }}
                    <button class=»submit»>CREER UN PATIENT</button>
            </form>
        </section>
    </div>
 </div>
{% endblock content %}
```

template --> form_patient.html

```
class PatientAddView(LoginRequiredMixin, CreateView):
    form_class = UserProfileForm
    model = User
    template_name = 'form_patient.html'

    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs['therapist'] = self.request.user.profile
        return kwargs

    def get_success_url(self):
        return reverse('index_speech')
```

views.py --> CreateView

```
class UserProfileForm(UserCreationForm):
    first_name = forms.CharField()
    last_name = forms.CharField()
    email = forms.EmailField(required=True)
    birth_date = forms.DateField()
    bio = forms.CharField(widget=forms.Textarea)
    review = forms.CharField(widget=forms.Textarea)
    profile_pic = forms.ImageField(required=False)
    pathology = forms.ModelChoiceField(queryset=Pathology.objects.all())
    num = forms.IntegerField()
    street = forms.CharField()
    zip_code = forms.CharField()
    city = forms.CharField()

    class Meta:
        model = User
        fields = ['first_name', 'last_name', 'email',
                  'username', 'password1', 'password2']

    def __init__(self, *args, therapist=None, **kwargs):
        super().__init__(*args, **kwargs)
        self.therapist = therapist

    def save(self, *args, **kwargs):
        user = super().save(*args, **kwargs)
        address = Address.objects.create(
            profile=user.profile, num=self.cleaned_data['num'], street=self.cleaned_data['street'],
            zip_code=self.cleaned_data['zip_code'], city=self.cleaned_data['city'])
        user.profile.birth_date = self.cleaned_data['birth_date']
        user.profile.therapist = self.therapist
        user.profile.bio = self.cleaned_data['bio']
        user.profile.review = self.cleaned_data['review']
        user.profile.profile_pic = self.cleaned_data['profile_pic']
        user.profile.pathology = self.cleaned_data['pathology']
        user.profile.save()
        user.profile.address.save()
        return user
```

forms.py -->UserCreationForm

Test unitaire et intégration

Test sur la création d'un profile patient

```
class UserProfileFormTest(TestCase):
    def setUp(self):
        self.pathology_id = Pathology.objects.create(name='Dyslexie').pk
        self.therapist = User.objects.create_user(
            username='Marieaumont', password='Therapist@25').pk
        return super().setUp()

    def test_userprofile_valid(self):
        form = UserProfileForm(data={
            'first_name': 'Lucas',
            'last_name': 'Desmarais',
            'email': 'helenedesmarais@email.com',
            'username': 'LucasD',
            'password1': '12test12',
            'password2': '12test12',
            'birth_date': '2013-05-12',
            'bio': 'Lucas est atteint de .....',
            'review': 'Bilan du patient .....',
            'profile_pic': 'lucasdesmarais.png',
            'pathology': self.pathology_id,
            'num': 10,
            'street': 'rue de la paix',
            'zip_code': 75015,
            'city': 'Paris',
        }, therapist=self.therapist)
        self.assertTrue(form.is_valid())

    def test_userprofile_missing_first_name(self):
        form = UserProfileForm(data={
            'first_name': '',
            'last_name': 'Desmarais',
            'email': 'helenedesmarais@email.com',
            'username': 'LucasD',
            'password1': '12test12',
            'password2': '12test12',
            'birth_date': '2013-05-12',
            'bio': 'Lucas est atteint de .....',
            'profile_pic': 'lucasdesmarais.png',
            'pathology': self.pathology_id,
            'num': 10,
            'street': 'rue de la paix',
            'zip_code': 75015,
            'city': 'Paris',
        }, therapist=self.therapist)
        self.assertFalse(form.is_valid())
        self.assertIn('first_name', form.errors)
```

test_forms.py

```
class UserProfileViewTest(TestCase):
    def setUp(self):
        self.user = get_user_model().objects.create_user(
        username='Marieaumont', password='Therapist@25').
pk
        self.patient = User.objects.create_user(
                username='ClarenceBr', password='12test12',
email='test@email.com')
                        self.pathology_id = Pathology.objects.
create(name='Dyslexie').pk
        self.form_url = reverse('form_patient')
        self.client = Client()
        self.patient = {
            'first_name': 'Clarence',
            'last_name': 'Brault',
            'username': 'ClarenceBr',
            'email': 'test@email.com',
            'password1': '12test12',
            'password2': '12test12',
            'birth_date': '12/05/2013',
            'bio': 'Clarence est atteint de .....',
            'profile_pic': '',
            'pathology': self.pathology_id,
            'num': '10',
            'street': 'rue de la paix',
            'zip_code': '75015',
            'city': 'Paris',
            'therapist': self.user
        }
        return super().setUp()

    def test_view_ok(self):
        login = self.client.login(
        username='Marieaumont', password='Therapist@25')
        self.assertTrue(login)
        response = self.client.get(self.form_url)
        self.assertEquals(response.status_code, 200)

    def test_form_ok(self):
        login = self.client.login(
        username='Marieaumont', password='Therapist@25')
        self.assertTrue(login)
        response = self.client.post(self.form_url, self.patient)
        self.assertEquals(response.status_code, 200)
        patient = User.objects.get(email='test@email.com')
        patient.pathology_id = self.pathology_id
        patient.first_name = 'Clarence'
```

test_views.py

Test fonctionnel

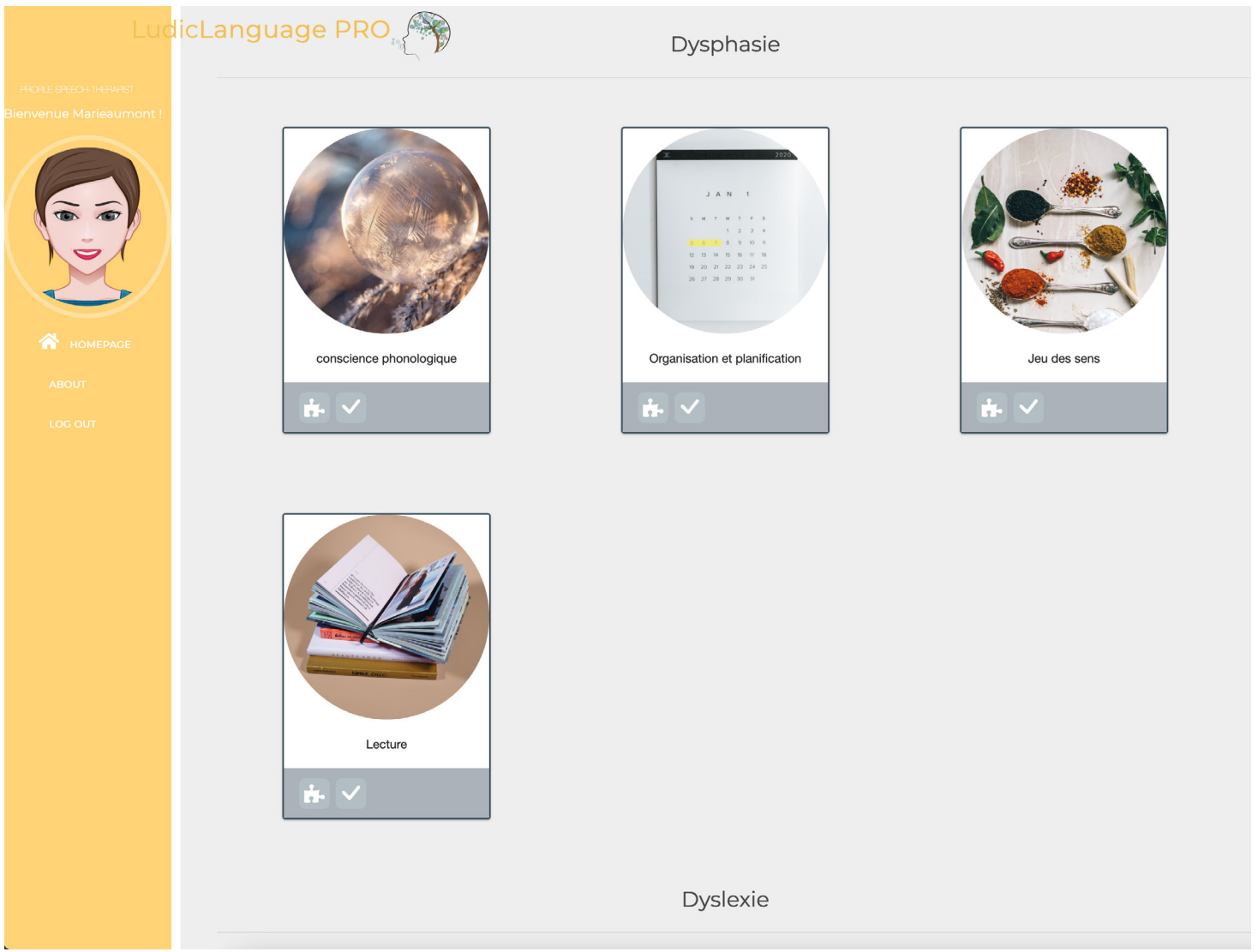
Test sur la création d'un profile patient

```
class LudicLanguageProfileTest(StaticLiveServerTestCase):

    def setUp(self):
        self.driver = webdriver.Firefox(options=firefox_options)
        self.user = get_user_model().objects.create_user(
            username='Marieaumont', first_name='Marie', password='12test12', email='test@email.com')
        self.user_id = self.user.pk
        self.user.save()
        self.therapist = Profile(user=self.user, birth_date='1975-05-12', state=2,
                                 bio='Marie est spécialisée .....', profile_pic='marieaumont.png'
                                 ).save()
        self.address = Address.objects.create(
            num=6, street='rue de ...', zip_code=92500, city='Paris', profile_id=self.user_id)
        self.patient = User.objects.create_user(
            username='LucasD', password='12test12', email='test@email.com')
        self.pathology_id = Pathology.objects.create(name='Dyslexie').pk
        Profile(user=self.patient, birth_date='2013-05-12', state=1,
                bio='Lucas est atteint de .....', profile_pic='lucasdesmarais.png',
                pathology_id=self.pathology_id, therapist_id=self.therapist).save()
        return super().setUp()

    def test_form(self):
        driver = self.driver
        driver.implicitly_wait(10)
        driver.get('%s%s' % (self.live_server_url, '/login/'))
        driver.find_element_by_id('id_username').send_keys('Marieaumont')
        driver.find_element_by_name('password').send_keys('12test12')
        driver.find_element_by_css_selector('.submit').click()
        driver.find_element_by_css_selector('.btn').click()
        driver.get('%s%s' % (self.live_server_url, '/form_patient/'))
        driver.find_element_by_name('first_name').send_keys('Clarence')
        driver.find_element_by_name('last_name').send_keys('Grey')
        driver.find_element_by_name('email').send_keys('helenegrey@email.com')
        driver.find_element_by_name('username').send_keys('ClarenceGr')
        driver.find_element_by_name('password1').send_keys('12test12')
        driver.find_element_by_name('password2').send_keys('12test12')
        driver.find_element_by_name('birth_date').send_keys('12/05/2015')
        driver.find_element_by_name('bio').send_keys('Clarence est un ....')
        driver.find_element_by_name('review').send_keys(
            'Clarence a besoin de  ....')
        driver.find_element_by_name('pathology').send_keys('Dyslexie')
        driver.find_element_by_name('num').send_keys('10')
        driver.find_element_by_name('street').send_keys('rue de ....')
        driver.find_element_by_name('zip_code').send_keys(92140)
        driver.find_element_by_name('city').send_keys('Clamart')
        driver.find_element_by_css_selector('.submit').click()
        driver.find_element_by_name('patient').click()
```

App Exercises

Liste des exercices par pathologies

- Templates



App Exercises

Liste des exercices en fonction des pathologies

- Template
- views.py

```
{% for pathology in pathology_list %}
        <div class=»row»>
          <div class=» d-flex  justify-content-center»>
            <h3>{{pathology.name}}</h3>
          </div>
          <hr>
            {% for exercise in pathology.exercise_set.all %}
              <div class=»col-lg-4 col-md-6 col-sm-12 d-flex  justify-
content-center»>
            <div class=’card exercise-card d-flex flex-column’>
            {% if exercise.picture1 %}
             <img src=»{{exercise.picture1.url}}» class=»card-img-top
exercise-img  rounded d-block align-m»>
              {% endif %}
           <p class=»card-title»>{{exercise.name}}</p>
          <div class=»card-footer d-flex exercise align-content-around
mt-auto»>
                  <a  class=»d-flex justify-content-center align-items-
center btn btn-detail»  type=»button» href=»{% url ‘form_ludicjourney’
exercise.id %}»
  data-toggle=»tooltip» data-placement=»bottom»>
<span>Ajouter l’exercice au parcours ludique </span>
<i class=»fas fa-puzzle-piece»></i></a>
<a  class=»d-flex justify-content-center align-items-center btn btn-
detail»  type=»button» href=»{% url ‘exercise_therapist’ %}»
data-toggle=»tooltip» data-placement=»bottom»>
<span>Liste des exercices</span>
<i class=»bi bi-check-lg»></i> </a>
</div>
</div>
</div>
{% endfor %}
        </div>
        {% endfor %}
```

template --> exercise_list.html

```
class ExerciseListView(LoginRequiredMixin,
ListView):
  template_name = ‘exercise_list.html’
  context_object_name = ‘pathology_list’
  model = Pathology

  def get_queryset(self):
    qs = super().get_queryset()
    qs.select_related(‘exercise’)
    return qs
```

views.py --> ListView **Version Django**

```
class ExerciseListView(LoginRequiredMixin,
ListView):
  template_name = ‘exercise_list.html’
  context_object_name = ‘exercise_list’
  model = Exercise

  def get_context_data(self, **kwargs):
    context = super().get_context_data()
    exercises = context[‘exercise_list’]
    pathologies = defaultdict(list)
    for exercise in exercises:
      pathologies[exercise.pathology.name].
append(exercise)
    path = sorted(pathologies.items())
    context[‘pathologies’] = path
    return context
```

views.py --> ListView **Version Python**

Test unitaire et intégration

Test sur l'affichage de la liste des exercices

```
class BaseTest(TestCase):
    def setUp(self):
        self.client = Client()
        self.factory = RequestFactory()
        self.user = get_user_model().objects.create_user(
            username='LucasD', first_name='Lucas', password='12test12', email='test@email.com')
        self.user_id = self.user.pk
        self.user.save()
        self.therapist = User.objects.create_user(
            username='Marieaumont', password='Therapist@25').pk

        self.pathology_id = Pathology.objects.create(
            name='Dyslexie', description=»la pathologie ..... «).pk
        self.profile = Profile(user=self.user, birth_date='2013-05-12', state=1,
                               bio='Lucas est atteint de ......', profile_pic='lucasdesmarais.png',
                               pathology_id=self.pathology_id, therapist_id=self.therapist).save()

        self.exercise = Exercise.objects.create(name='Améliorer ....', description=»L'aider ...», picture1='Winter.png',
                                                picture2=»puzzle.png», pathology_id=self.pathology_id, therapist_id=self.therapist,
                                                description_game=»Rassemble ...», title_game='Puzzle').pk
        return super().setUp()

class ExerciseListTest(BaseTest):
    def setUp(self):
        self.list_url = reverse('exercise_list')
        return super().setUp()

    def test_view_ok(self):
        login = self.client.login(
            username='Marieaumont', password='Therapist@25')
        self.assertTrue(login)

        response = self.client.get(self.list_url)
        assert response.status_code == 200

    def test_queryset(self):
        request = self.factory.get(self.list_url)
        view = ExerciseListView()
        view.request = request
        qs = view.get_queryset()
        self.assertQuerysetEqual(qs, Pathology.objects.all())
```

test_views.py

Test fonctionnel

Test sur l'affichage de la liste des exercices

```
class LudicLanguageExerciseTest(StaticLiveServerTestCase):
    def setUp(self):
        self.driver = webdriver.Firefox(options=firefox_options)
        self.user = get_user_model().objects.create_user(
            username='Marieaumont', first_name='Marie', password='12test12', email='test@email.com')
        self.user_id = self.user.pk
        self.user.save()
        self.therapist = Profile(user=self.user, birth_date='1975-05-12', state=2,
                                 bio='Marie est spécialisée .....', profile_pic='marieaumont.png'
                                 ).save()
        self.pathology_id = Pathology.objects.create(name='Dyslexie').pk

        self.exercise = Exercise.objects.create(name='Améliorer ....', description=»L'aider ...», picture1='Winter.png',
                                                picture2=»puzzle.png», pathology_id=self.pathology_id, therapist_id=self.therapist,
                                                description_game=»Rassemble ...», title_game='Puzzle').pk
        return super().setUp()

    def test_exercise_list(self):
        driver = self.driver
        driver.implicitly_wait(10)
        driver.get('%s%s' % (self.live_server_url, '/login/'))
        driver.find_element_by_id('id_username').send_keys('Marieaumont')
        driver.find_element_by_name('password').send_keys('12test12')
        driver.find_element_by_css_selector('.submit').click()
        driver.find_element_by_name('ludic_journey').click()
```

shell_plus

Console shell_plus

Pour tester les requêtes, j'ai installé **django_extensions**

Pour débugger le code, j'ai installé et utilisé **IPDB**

```
from ludic_language.workshops.models import Workshop
# Shell Plus Django Imports
from django.core.cache import cache
from django.conf import settings
from django.contrib.auth import get_user_model
from django.db import transaction
from django.db.models import Avg, Case, Count, F, Max, Min, Prefetch, Q, Sum, When
from django.utils import timezone
from django.urls import reverse
from django.db.models import Exists, OuterRef, Subquery
Python 3.9.9 (v3.9.9:ccb0e6a345, Nov 15 2021, 13:29:20)
Type 'copyright', 'credits' or 'license' for more information
IPython 8.1.1 -- An enhanced Interactive Python. Type '?' for help.

[In [1]]: User.objects.all()
Out[1]: <QuerySet [<User: admin>, <User: Madelinefranklin>, <User: XavierDam>, <User: FredericJe>, <User: RoseMc>, <User: PatrickPe>, <User: AgatheSa>, <User: CharlotteBr>, <User: MarcusGa>, <User: ClarenceGr>, <User: MaggieSe>, <User: MarthaEs>, <User: Marieaumont>, <User: LilahBa>, <User: LucasDes>]>

[In [2]]: Profile.objects.filter(Q(address__city__icontains=search) | Q(address__zip_code__icontains=search))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [2], in <cell line: 1>()
----> 1 Profile.objects.filter(Q(address__city__icontains=search) | Q(address__zip_code__icontains=search))

NameError: name 'search' is not defined

[In [3]]: Profile.objects.filter(Q(address__city__icontains=Paris) | Q(address__zip_code__icontains=75))
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [3], in <cell line: 1>()
----> 1 Profile.objects.filter(Q(address__city__icontains=Paris) | Q(address__zip_code__icontains=75))

NameError: name 'Paris' is not defined

[In [4]]: Profile.objects.filter(Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
Out[4]: <QuerySet [<Profile: XavierDam>, <Profile: Marieaumont>, <Profile: RoseMc>]>

[In [5]]: Profile.objects.filter(Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75).filter(profile__state=2))
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [5], in <cell line: 1>()
----> 1 Profile.objects.filter(Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75).filter(profile__state=2))

AttributeError: 'Q' object has no attribute 'filter'

[In [6]]: Profile.objects.filter(profile__user=2) Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
  Input In [6]
    Profile.objects.filter(profile__user=2) Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
                                            ^
SyntaxError: invalid syntax

[In [7]]: Profile.objects.filter(state=2), Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
  Input In [7]
    Profile.objects.filter(state=2), Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
                                                                         ^
SyntaxError: unmatched ')'

[In [8]]: Profile.objects.filter(state=2), Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75)
Out[8]:
(<QuerySet [<Profile: Marieaumont>, <Profile: PatrickPe>, <Profile: MarthaEs>, <Profile: RoseMc>, <Profile: Madelinefranklin>, <Profile: FredericJe>, <Profile: XavierDam>]>,
 <Q: (OR: ('address__city__icontains', 'Paris'), ('address__zip_code__icontains', 75))>)

[In [9]]: Profile.objects.filter(state=2, Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
  Input In [9]
    Profile.objects.filter(state=2, Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75))
                                    ^
SyntaxError: positional argument follows keyword argument

[In [10]]: Profile.objects.filter(Q(address__city__icontains='Paris') | Q(address__zip_code__icontains=75), state=2)
Out[10]: <QuerySet [<Profile: XavierDam>, <Profile: Marieaumont>, <Profile: RoseMc>]>

[In [11]]:
```

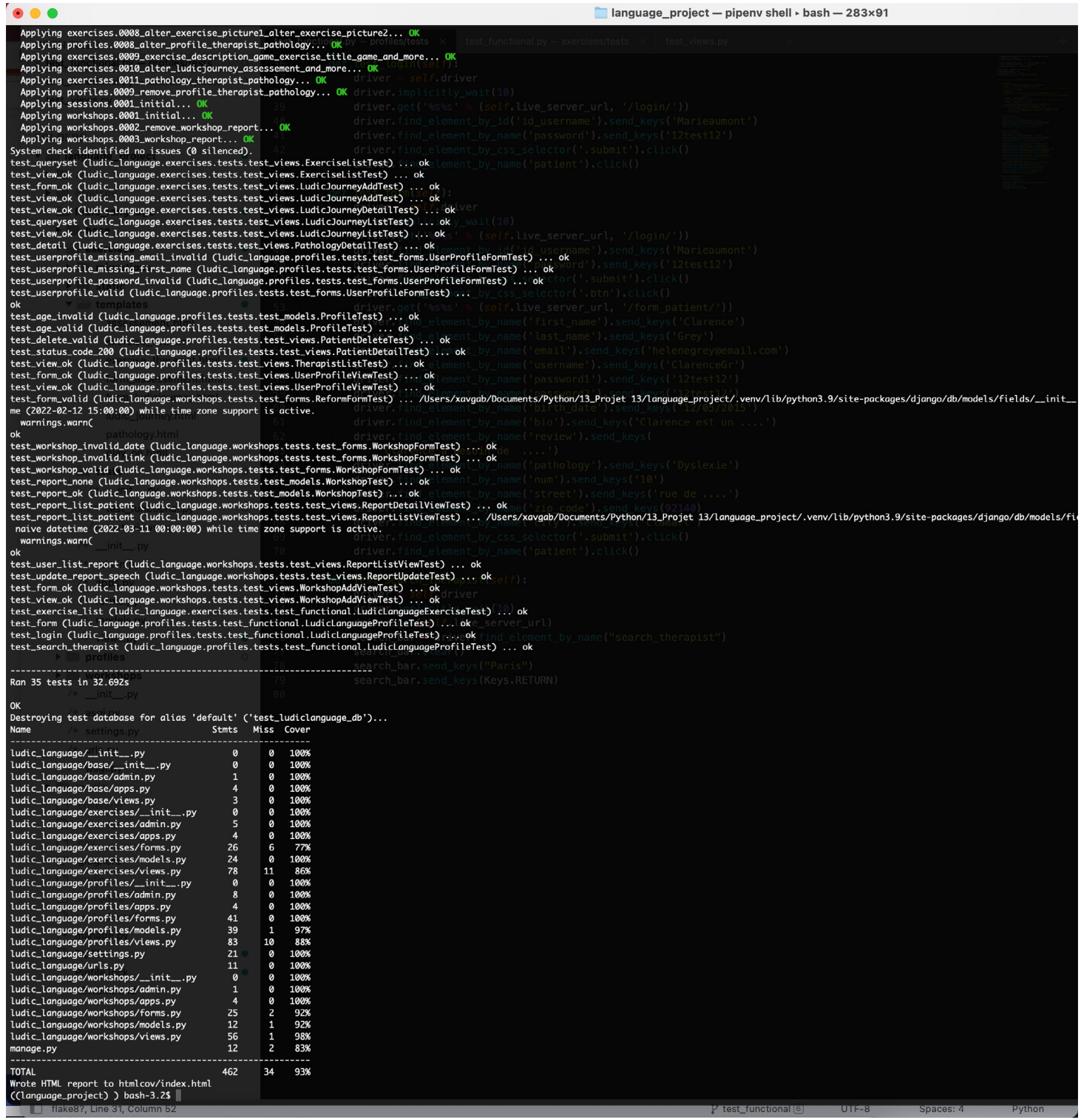
Couverture des tests

Couverture de test à 93%

coverage run manage.py test -v 2 && covert report && html

Coverage report: 93%

Module	statements	missing	excluded	coverage
ludic_language/__init__.py	0	0	0	100%
ludic_language/base/__init__.py	0	0	0	100%
ludic_language/base/admin.py	1	0	0	100%
ludic_language/base/apps.py	4	0	0	100%
ludic_language/base/views.py	3	0	0	100%
ludic_language/exercises/__init__.py	0	0	0	100%
ludic_language/exercises/admin.py	5	0	0	100%
ludic_language/exercises/apps.py	4	0	0	100%
ludic_language/exercises/forms.py	26	6	0	77%
ludic_language/exercises/models.py	24	0	0	100%
ludic_language/exercises/views.py	78	11	0	86%
ludic_language/profiles/__init__.py	0	0	0	100%
ludic_language/profiles/admin.py	8	0	0	100%
ludic_language/profiles/apps.py	4	0	0	100%
ludic_language/profiles/forms.py	41	0	0	100%
ludic_language/profiles/models.py	39	1	0	97%
ludic_language/profiles/views.py	83	10	0	88%
ludic_language/settings.py	21	0	0	100%
ludic_language/urls.py	11	0	0	100%
ludic_language/workshops/__init__.py	0	0	0	100%
ludic_language/workshops/admin.py	1	0	0	100%
ludic_language/workshops/apps.py	4	0	0	100%
ludic_language/workshops/forms.py	25	2	0	92%
ludic_language/workshops/models.py	12	1	0	92%
ludic_language/workshops/views.py	56	1	0	98%
manage.py	12	2	0	83%
Total	**462**	**34**	**0**	**93%**



Ludic Language

Migration des données

Migration des données

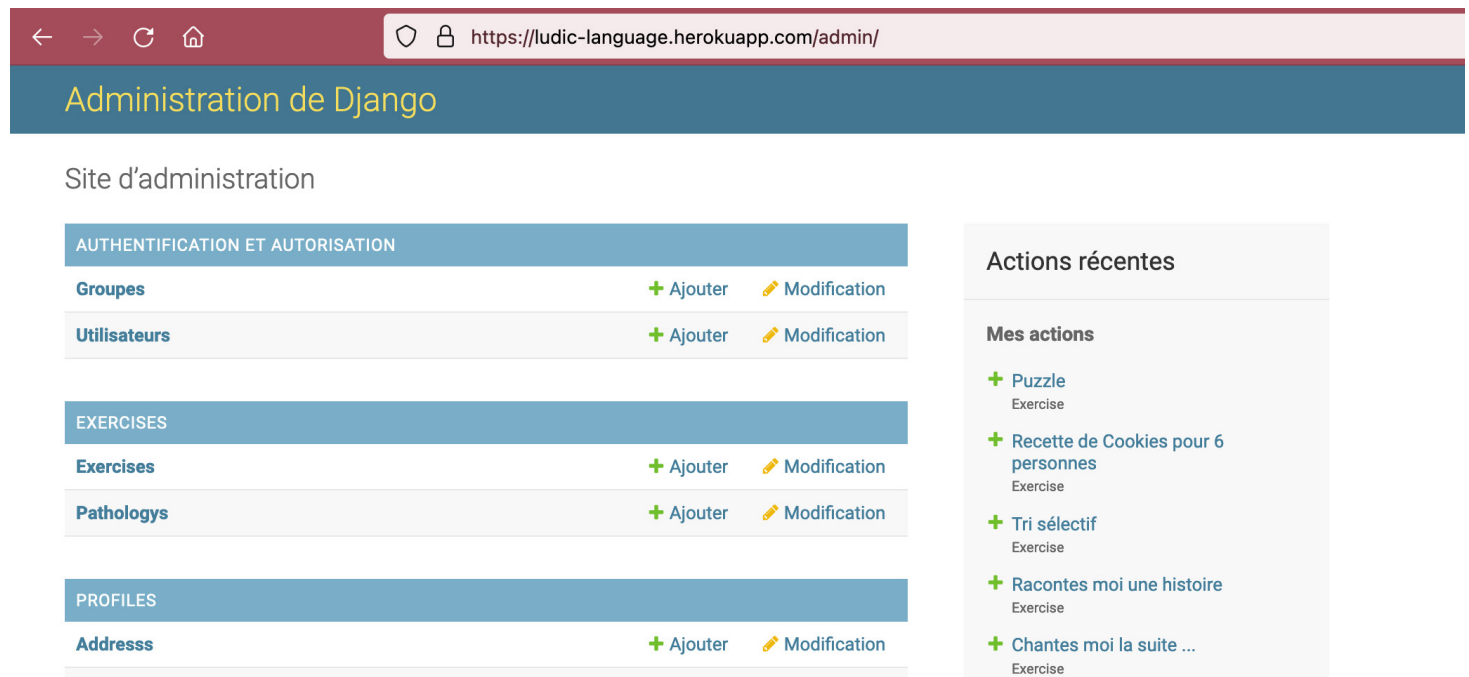
Création des données via la plateforme Admin Django
python manage.py makemigrations --empty exercises
python manage.py migrate
Heroku run bash
python manage.py migrate

```
def create_pathologies(apps, schema_editor):
    Pathology = apps.get_model('exercises', 'Pathology')
    Pathology.objects.get_or_create(name='Dyslexie', description='Troubles ....')


def reverse_pathologies(apps, schema_editor):
    pass


class Migration(migrations.Migration):

    dependencies = [
        ('exercises', '0011_pathology_therapist_pathology'),
    ]

    operations = [
        migrations.RunPython(create_pathologies, reverse_pathologies),
    ]
```

migrations --> 0012_auto_44455466.py

Admin Django

Création des données via la plateforme Admin Django
Heroku



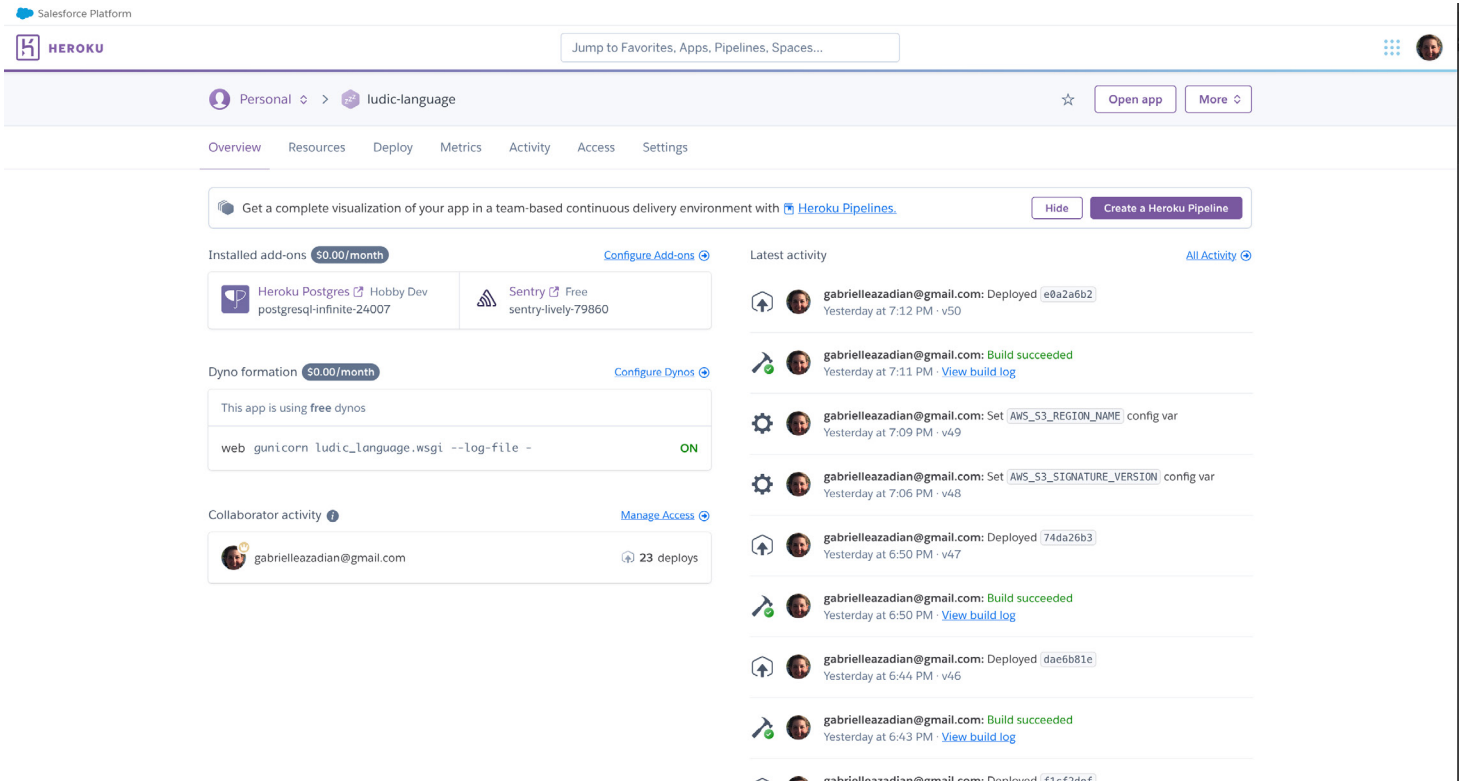
Déploiement

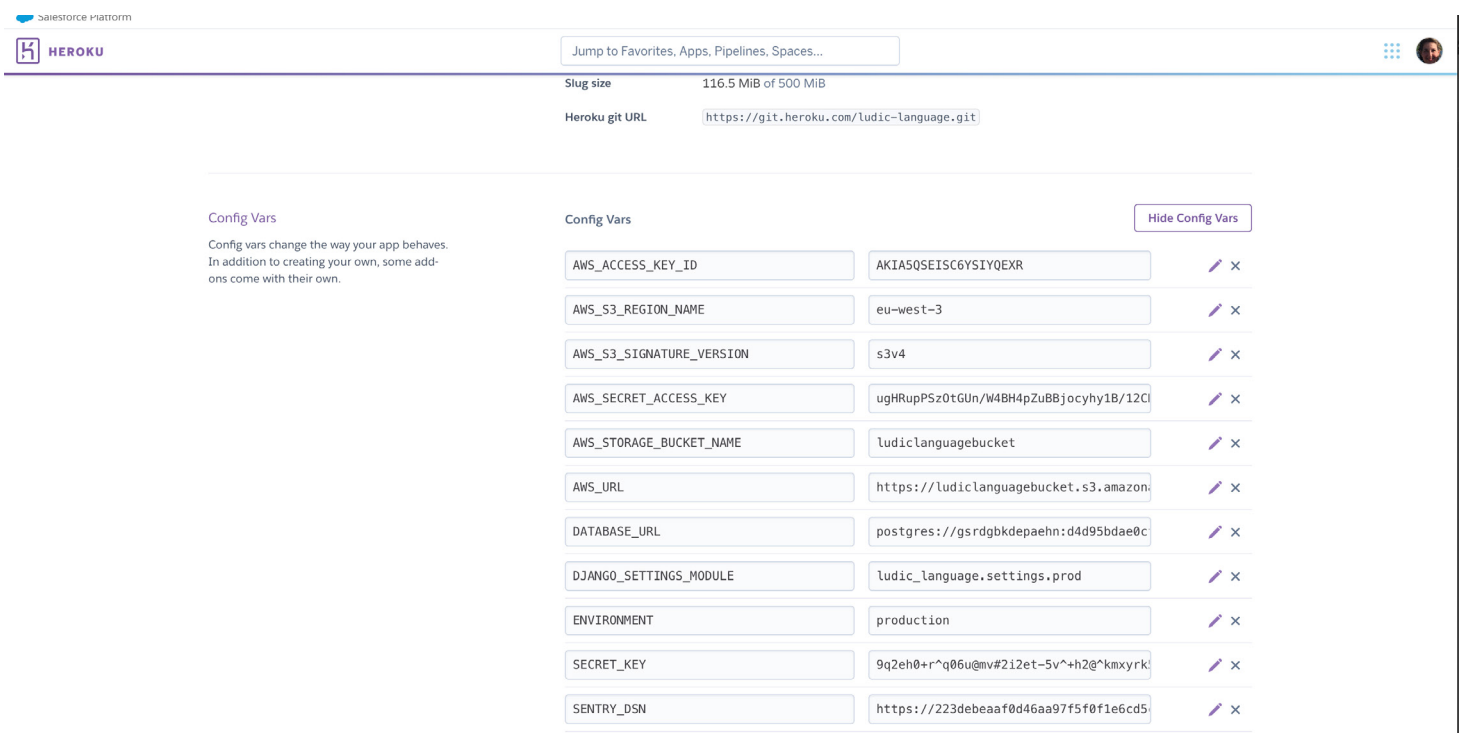
Settings --> prod.py

```
SECRET_KEY = os.getenv(«SECRET_KEY»)
sentry_sdk.init(
    «https://223debeaaf0d46aa97f5f0f1e6cd5c45@o1176728.ingest.sentry.io/6274790»,

    # Set traces_sample_rate to 1.0 to capture 100%
    # of transactions for performance monitoring.
    # We recommend adjusting this value in production.
    traces_sample_rate=1.0
)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
INSTALLED_APPS = INSTALLED_APPS + [‘storages’, ]
DEBUG = False
ALLOWED_HOSTS = [‘.herokuapp.com’]
DATABASES = {
    ‘default’: {
        ‘ENGINE’: ‘django.db.backends.postgresql’,
        ‘NAME’: os.getenv(«DB_NAME»),
        ‘USER’: os.getenv(«DB_USER»),
        ‘PASSWORD’: os.getenv(«DB_PASSWORD»),
        ‘HOST’: os.getenv(«DB_HOST»),
        ‘PORT’: ‘5432’
    }
}
# Amazon S3
AWS_ACCESS_KEY_ID = os.getenv(«AWS_ACCESS_KEY_ID»)
AWS_SECRET_ACCESS_KEY = os.getenv(«AWS_SECRET_ACCESS_KEY»)
AWS_STORAGE_BUCKET_NAME = os.getenv(«AWS_STORAGE_BUCKET_NAME»)

AWS_S3_REGION_NAME = os.getenv(«AWS_S3_REGION_NAME»)
AWS_S3_SIGNATURE_VERSION = os.getenv(«AWS_S3_SIGNATURE_VERSION»)
AWS_S3_FILE_OVERWRITE = False
AWS_DEFAULT_ACL = None

DEFAULT_FILE_STORAGE = ‘storages.backends.s3boto3.S3Boto3Storage’

# Activate Django-Heroku.
django_heroku.settings(locals())
```

settings --> prod.py

Heroku


Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal > ludic-language
Open app
More
Overview
Resources
Deploy
Metrics
Activity
Access
Settings
Get a complete visualization of your app in a team-based continuous delivery environment with Heroku Pipelines.
Hide
Create a Heroku Pipeline
Installed add-ons $0.00/month
Configure Add-ons
Heroku Postgres Hobby Dev
postgresql-infinite-24007
Sentry Free
sentry-lively-79860
Dyno formation $0.00/month
Configure Dynos
This app is using free dynos
web gunicorn ludic_language.wsgi --log-file - ON
Collaborator activity
Manage Access
gabrielleazadian@gmail.com
23 deploys
Latest activity
All Activity
gabrielleazadian@gmail.com: Deployed e0a2a6b2
Yesterday at 7:12 PM · v50
gabrielleazadian@gmail.com: Build succeeded
Yesterday at 7:11 PM · View build log
gabrielleazadian@gmail.com: Set AWS_S3_REGION_NAME config var
Yesterday at 7:09 PM · v49
gabrielleazadian@gmail.com: Set AWS_S3_SIGNATURE_VERSION config var
Yesterday at 7:06 PM · v48
gabrielleazadian@gmail.com: Deployed 74da26b3
Yesterday at 6:50 PM · v47
gabrielleazadian@gmail.com: Build succeeded
Yesterday at 6:50 PM · View build log
gabrielleazadian@gmail.com: Deployed dae6b81e
Yesterday at 6:44 PM · v46
gabrielleazadian@gmail.com: Build succeeded
Yesterday at 6:43 PM · View build log


Salesforce Platform
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Slug size 116.5 MiB of 500 MiB
Heroku git URL https://git.heroku.com/ludic-language.git
Config Vars
Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.
Config Vars
Hide Config Vars
AWS_ACCESS_KEY_ID AKIA5QSEISC6YSIYQEXR
AWS_S3_REGION_NAME eu-west-3
AWS_S3_SIGNATURE_VERSION s3v4
AWS_SECRET_ACCESS_KEY ugHRupPSzOtGUn/W4BH4pZuBBjocyhy1B/12Cl
AWS_STORAGE_BUCKET_NAME ludiclanguagebucket
AWS_URL https://ludiclanguagebucket.s3.amazon
DATABASE_URL postgres://gsrdgbkdepaehn:d4d95bdae0c
DJANGO_SETTINGS_MODULE ludic_language.settings.prod
ENVIRONMENT production
SECRET_KEY 9q2eh0+r^q06u@mv#2i2et-5v^+h2@^kmxyrk
SENTRY_DSN https://223debeaaf0d46aa97f5f0f1e6cd5

Heroku -->plateforme déployée