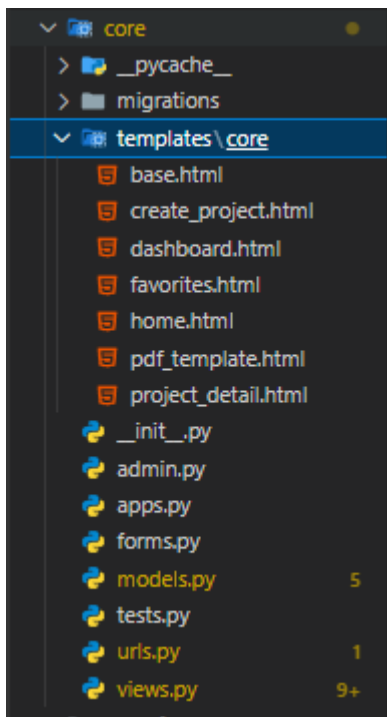


GameForge – Générateur de jeux vidéo par IA

Mon rôle dans le projet était d'être le développeur fullstack, coder la partie qui concerne le VIEWS du coup, créer les fonctions qui traitent les données et les envoyer aux pages HTML dans le dossier templates et du coup aussi coder les pages HTML afin d'avoir un bon rendu de la page à afficher au client.



La hiérarchie des dossiers du projet que j'ai codée en collaboration avec les autres membres de l'équipe

- models.py : (vide) emplacement pour les modèles personnalisés.
- views.py : contient la vue signup_view.
- urls.py : routes login, logout, signup.
- templates/accounts/login.html : template de connexion.
- templates/accounts/signup.html — template d'inscription.

```
view  GO  Run  Terminal  Help
views.py 9+  dashboard.html X
core > templates > core > dashboard.html > div#loader.loader
1  {% extends "core/base.html" %}
2  {% block title %}Mon tableau de bord – GameForge{% endblock %}
3  {% block content %}
4  <div class="panel">
5      <div class="panel-head">
6          <h2>Mes projets</h2>
7          <div>
8              <a class="btn" href="{% url 'core:create' %}">Nouveau projet</a>
9              <a class="btn ghost" href="{% url 'core:explore_free' %}">Exploration libre</a>
10         </div>
11     </div>
12     <table class="table">
13         <thead><tr><th>Titre</th><th>Genre</th><th>Public</th><th>Actions</th></tr></thead>
14         <tbody>
15             {% for p in projects %}
16             <tr>
17                 <td><a href="{{ p.get_absolute_url }}">{{ p.title }}</a></td>
18                 <td>{{ p.genre }}</td>
19                 <td>{{ p.is_public|yesno:"Oui,Non" }}</td>
20                 <td>
21                     <button class="btn primary gen-btn" data-pid="{{ p.id }}">Générer IA</button>
22                     <a class="btn ghost" href="{% url 'core:export_pdf' p.slug %}">PDF</a>
23                 </td>
24             </tr>
25             {% empty %}
26             <tr><td colspan="4" class="muted">Aucun projet pour l'instant.</td></tr>
27             {% endfor %}
28         </tbody>
29     </table>
30 </div>
31
32 <div id="loader" class="loader" hidden>
33     <div class="spinner"></div>
34     <div>Génération en cours.</div>
35 </div>
36
37 <script>
38 document.querySelectorAll(".gen-btn").forEach(btn=>{
39     btn.addEventListener("click", async () => {
40         const pid = btn.dataset.pid;
41         const loader = document.getElementById("loader");
42         loader.hidden = false;
43         const resp = await fetch("{% url 'core:generate' %}", {
44             method: "POST",
45             headers: {"Content-Type": "application/json", "X-CSRFToken": "{{ csrf_token }}"},
46             body: JSON.stringify({project_id: pid})
47         });
48         const data = await resp.json();
49         loader.hidden = true;
50         if(data.status === "ok"){ window.location.href = data.project_url; }
51         else{ alert(data.error || "Erreur inconnue."); }
52     });
53 });
54 </script>
55 {% endblock %}
```

Un exemple d'une page HTML que l'on a structurée avec une fonction JavaScript qui permet aussi de dynamiser la page, les données sont récupérées par les VIEWS et à travers les variables, on les injecte dans l'HTML

```

User = get_user_model()

class GameProject(models.Model):
    author = models.ForeignKey(User, on_delete=models.CASCADE, related_name="projects")
    title = models.CharField(max_length=200)
    slug = models.SlugField(max_length=220, unique=True, blank=True)
    genre = models.CharField(max_length=100)
    ambiance = models.CharField(max_length=200, blank=True)
    keywords = models.TextField(blank=True)
    references = models.TextField(blank=True)
    is_public = models.BooleanField(default=False)
    generated = models.JSONField(null=True, blank=True)
    image_character = models.ImageField(upload_to="generated/characters/", null=True, blank=True)
    image_environment = models.ImageField(upload_to="generated/environments/", null=True, blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)

    def save(self, *args, **kwargs):
        if not self.slug:
            base = slugify(self.title)[:150]
            self.slug = f"{base}-{int(time.time())}"
        super().save(*args, **kwargs)

    def get_absolute_url(self):
        return reverse("core:project_detail", args=[self.slug])

    def __str__(self):
        return f"{self.title} - {self.author.username}"

class Favorite(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE, related_name="favorites")
    project = models.ForeignKey(GameProject, on_delete=models.CASCADE, related_name="favorited_by")
    created_at = models.DateTimeField(auto_now_add=True)

    class Meta:
        unique_together = ("user", "project")

class ApiUsage(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    day_key = models.CharField(max_length=10) # format YYYYMMDD
    count = models.IntegerField(default=0)

    class Meta:
        unique_together = ("user", "day_key")

```

Ici on voit les modèles qui nous permettent de structurer et créer une passerelle entre les données entrées par le front et le formulaire vers la base de données.

On voit que l'on définit les données que l'on a pour chaque objet et le type

Les patterns nous permet de définir les chemin url que nous permet de changer page html et du coup afficher une View different.

```
urlpatterns = [
    path("", views.HomeView.as_view(), name="home"),
    path("search/", views.search_view, name="search"),
    path("dashboard/", views.DashboardView.as_view(), name="dashboard"),
    path("create/", views.CreateProjectView.as_view(), name="create"),
    path("project/<slug:slug>/", views.ProjectDetailView.as_view(), name="project_detail"),
    path("favorite/<slug:slug>/", views.toggle_favorite, name="toggle_favorite"),
    path("favorites/", views.favorites_view, name="favorites"),
    path("export/<slug:slug>/pdf/", views.export_project_pdf, name="export_pdf"),

    # IA
    path("generate/", views.generate_game_view, name="generate"),          # POST JSON {project_id}
    path("explore/", views.explore_free_view, name="explore_free"),        # GET -> crée & génère aléatoire
]
```

```

6
7 class HomeView(ListView):
8     model = GameProject
9     template_name = "core/home.html"
10    context_object_name = "projects"
11    paginate_by = 12
12    def get_queryset(self):
13        return GameProject.objects.filter(is_public=True).order_by("-created_at")
14
15    def search_view(request):
16        q = request.GET.get("q", "")
17        projects = GameProject.objects.filter(is_public=True).filter(
18            Q(title__icontains=q) | Q(genre__icontains=q) | Q(keywords__icontains=q) | Q(ambiance__icontains=q)
19        ).order_by("-created_at")
20        return render(request, "core/home.html", {"projects": projects, "search": q})
21
22 class DashboardView(LoginRequiredMixin, ListView):
23     model = GameProject
24     template_name = "core/dashboard.html"
25     context_object_name = "projects"
26    def get_queryset(self):
27        return GameProject.objects.filter(author=self.request.user).order_by("-created_at")
28
29 class ProjectDetailView(DetailView):
30     model = GameProject
31     template_name = "core/project_detail.html"
32     slug_field = "slug"
33
34 class CreateProjectView(LoginRequiredMixin, CreateView):
35     model = GameProject
36     form_class = ProjectCreateForm
37     template_name = "core/create_project.html"
38    def form_valid(self, form):
39        obj = form.save(commit=False)
40        obj.author = self.request.user
41        obj.save()
42        return redirect(obj.get_absolute_url())
43
44    def _day_key():
45        now = datetime.datetime.utcnow()
46        return now.strftime("%Y%m%d")
47
48    def _check_quota(user):
49        if not user.is_authenticated:
50            return False, "Authentification requise."
51        day = _day_key()
52        usage, _ = ApiUsage.objects.get_or_create(user=user, day_key=day)
53        if usage.count >= settings.DAILY_GENERATION_LIMIT:
54            return False, f"Limite quotidienne atteinte ({settings.DAILY_GENERATION_LIMIT}). Réessaie demain."
55        usage.count += 1
56        usage.save()
57        return True, ""

```

- Schéma de flux des données

