Dossier de présentation – Projets techniques

PennyPal - Application de gestion budgétaire

PennyPal est une application fullstack sécurisée développée en .NET 8 (C#) et React (TypeScript), permettant de gérer un budget personnel. Elle met en œuvre une architecture claire avec authentification robuste (JWT, refresh tokens, rotation, cookies HttpOnly), des règles métiers isolées, et un déploiement via Docker et NGINX.

L'application est disponible en ligne avec des comptes de démonstration permettant une prise en main immédiate.

Stack technique :

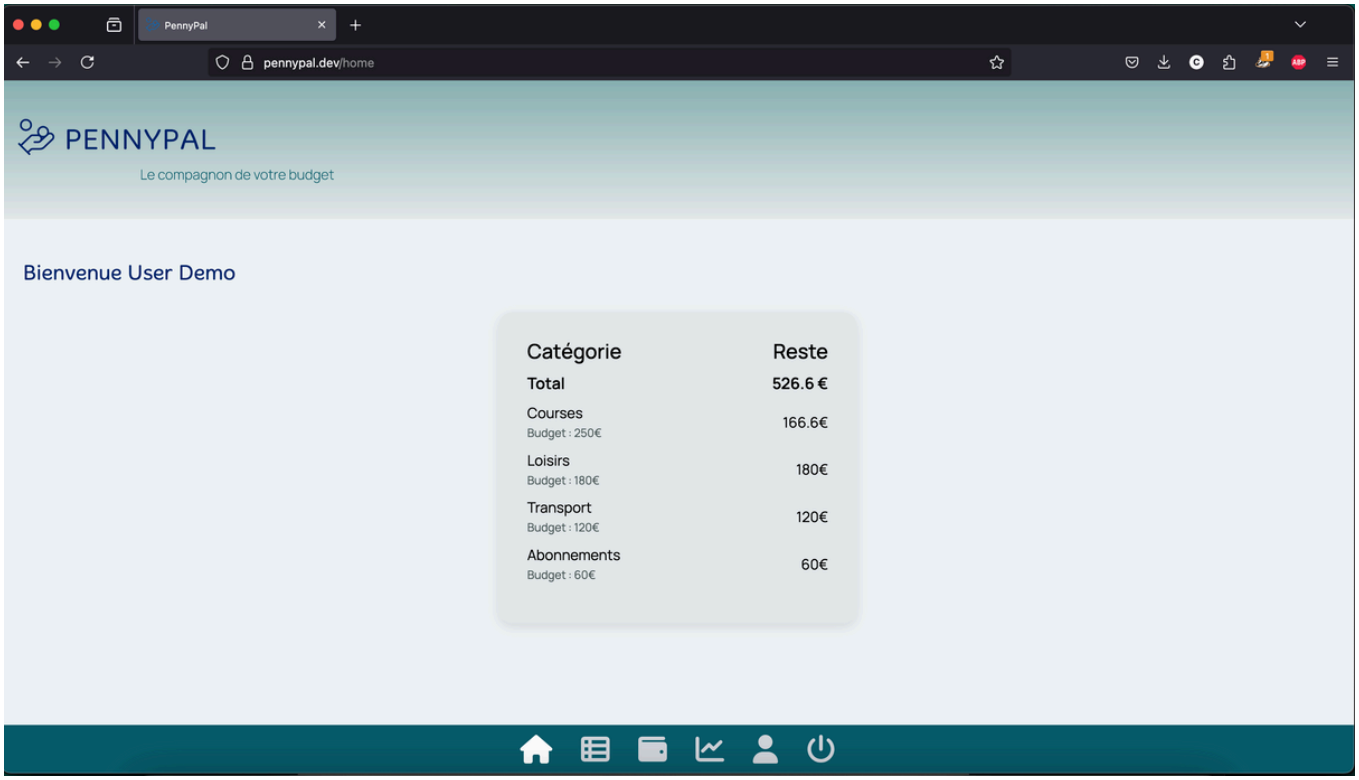
- C#/.NET 8 · React / TypeScript · SQL Server
- Authentification sécurisée : JWT + refresh tokens + cookies HttpOnly
- Docker · Docker Compose · NGINX

Liens utiles :

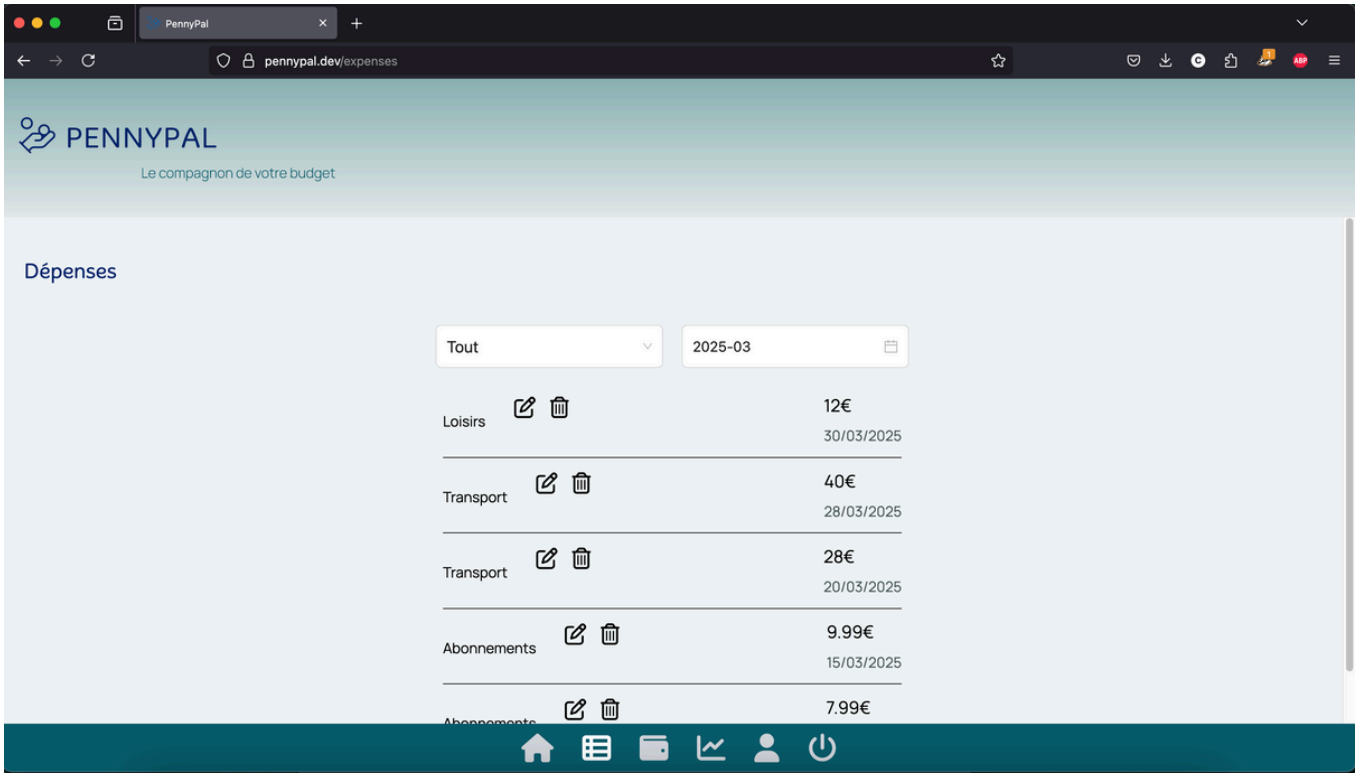
- Démo : https://www.pennypal.dev
- GitHub : https://github.com/CamilleNerriere/PennyPal

Me contacter pour accéder aux comptes de démonstration à l'adresse suivante : camille.nerriere@proton.me

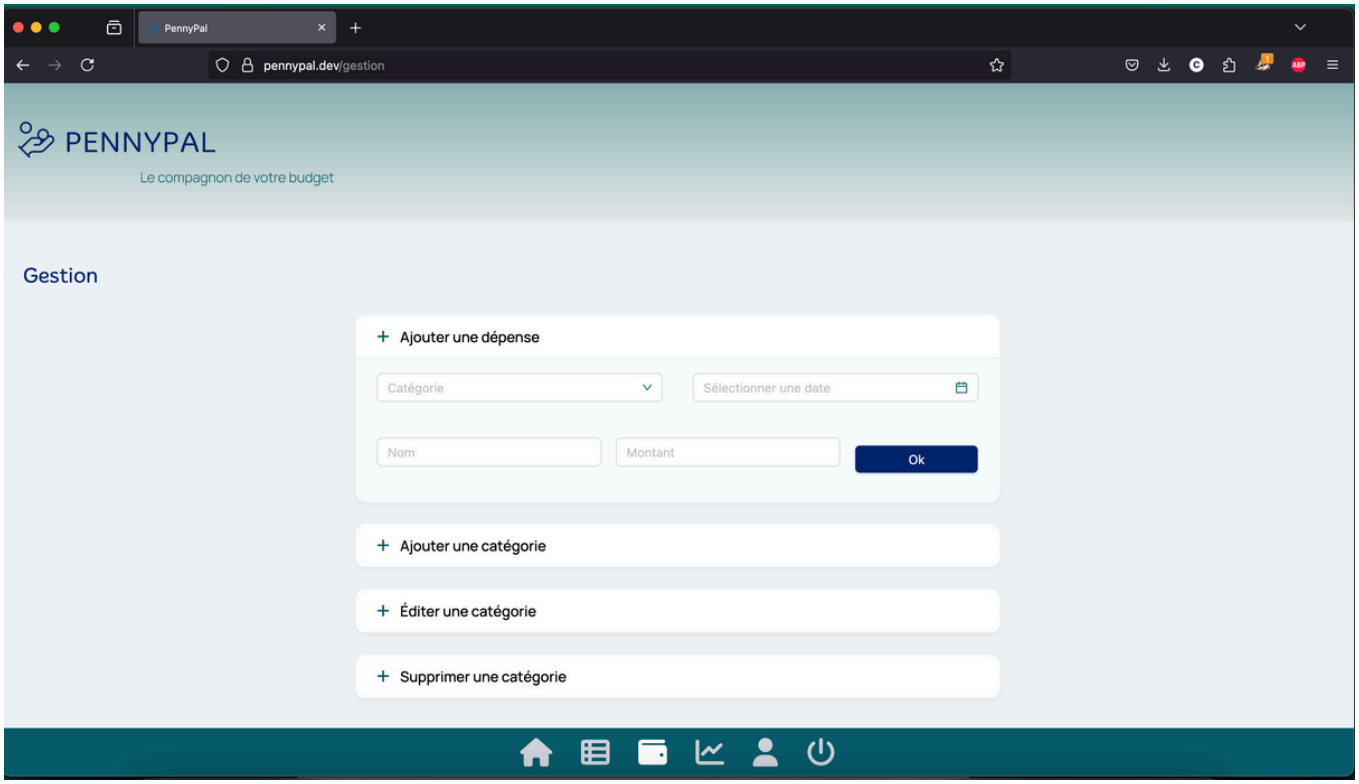
Captures d’écran



Page d’accueil - utilisateur connecté



Page visualisation/édition/suppression des dépenses



Page de gestion

Extraits de code

Authentification sécurisée et refresh token via cookies sécurisés

```
[HttpPost("RefreshToken")]
0 references
public async Task<IActionResult> RefreshToken()
{
    var refreshToken = Request.Cookies["refreshToken"];

    if (string.IsNullOrEmpty(refreshToken))
    {
        throw new Unauthorized(401, "Invalid Token");
    }

    var (newAccessToken, newRefreshToken, refreshExpiry) = await _authService.RefreshToken(refreshToken);

    Response.Cookies.Append("refreshToken", newRefreshToken, new CookieOptions
    {
        HttpOnly = true,
        Secure = _env.IsProduction(),
        SameSite = _env.IsProduction() ? SameSiteMode.Strict : SameSiteMode.Lax,
        Expires = refreshExpiry
    }
    );

    return Ok(new { token = newAccessToken });
}
```

Ce contrôleur expose l'endpoint /RefreshToken. Il lit le token depuis le cookie sécurisé, vérifie sa présence, puis délègue au service. Il applique les bonnes pratiques de sécurité web contre les attaques XSS / CSRF.

```
2 references
public async Task<(string accessToken, string refreshToken, DateTime refreshExpiry)> RefreshToken(string OldRefreshToken)
{
    var token = await _refreshTokenRepository.GetByToken(OldRefreshToken);

    if (token == null || token.Expires < DateTime.UtcNow || token.Revoked || token.SessionExpiresAt < DateTime.UtcNow)
    {
        _logger.LogWarning("Invalid refresh token attempt (partial): {TokenPrefix}...", OldRefreshToken[..10]);
        throw new Unauthorized(401, "Invalid Token");
    }

    await _refreshTokenRepository.InvalidateToken(token);

    var newRefreshToken = _authHelper.GenerateRefreshToken();

    var now = DateTime.UtcNow;
    var sessionLimit = token.SessionExpiresAt;
    var refreshExpiry = now.AddMinutes(15) > sessionLimit ? sessionLimit : now.AddMinutes(15);

    var newToken = new RefreshToken
    {
        Token = newRefreshToken,
        Expires = refreshExpiry,
        CreatedAt = DateTime.UtcNow,
        CreatedByIp = _authHelper.GetClientIp(),
        UserId = token.UserId,
        ReplacedByToken = token.Token,
        SessionExpiresAt = token.SessionExpiresAt
    };

    await _refreshTokenRepository.AddToken(newToken);

    var accessToken = _authHelper.CreateToken(token.UserId);

    return (accessToken, newRefreshToken, refreshExpiry);
}
```

Ce service vérifie la validité du token, révoque l'ancien, et crée un nouveau token limité par la session. Il implémente une logique défensive robuste, limitant la durée de vie des sessions même en cas de token encore valide.

PDF & Audio Synchronizer

Outil Python permettant de synchroniser un fichier audio avec le contenu textuel d'un PDF, ligne par ligne et page par page. Il repose sur une transcription automatique de l'audio (Vosk), une analyse structurelle du PDF (coordonnées des lignes) et un algorithme de matching hybride.

Points forts :

- Extraction des coordonnées textuelles dans le PDF pour une localisation fine
- Algorithme de matching audio ↔ texte avec marge d'erreur sur la transcription et pondération croisée : proximité temporelle et proximité spatiale dans la même page, pour éviter les faux positifs et garantir la précision

Stack technique :

- Python 3.13 · FastAPI · Vosk · PyMuPDF · ffmpeg · Tkinter

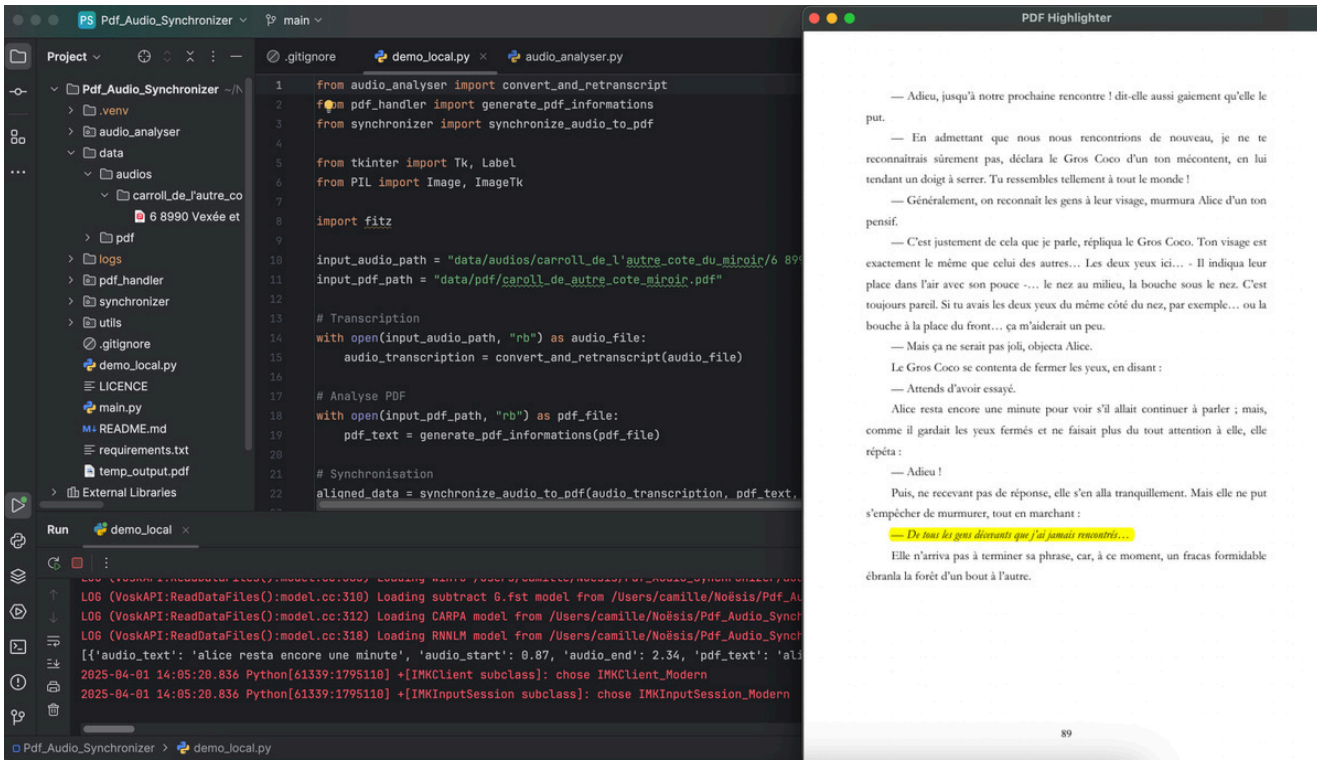
Liens utiles :

- GitHub : https://github.com/CamilleNerriere/PDF_Audio_Synchronizer

Extrait d'un dictionnaire contenant les données de synchronisation :

```
{
    'audio_text': 'alice resta encore une minute',
    'audio_start': 0.87, 'audio_end': 2.34,
    'pdf_text': 'alice resta encore une minute pour voir s'il allait
continuer à parler  mais', '
    pdf_coords': [99.23995971679688, 425.2620849609375,
527.9911499023438,              440.9755859375],
    'page_num': 89, 'line_index': 15
}
```

Capture d’écran



Aperçu du pdf surligné de manière synchronisée

Extraits de code

```python
def preprocess_text(text):  2 usages  CamilleNerriere
    text = re.sub( pattern: r'[^\w\s]',  repl: '', text.lower())
    text = text.replace( __old: '-',  __new: ' ')
    return ' '.join(text.split())  # Normalize whitespace


def calculate_text_similarity(text1, text2):  1 usage  CamilleNerriere
    p_text1 = preprocess_text(text1)
    p_text2 = preprocess_text(text2)

    return fuzz.partial_ratio(p_text1, p_text2)
```

```python
def synchronize_audio_to_pdf(audio_transcription, pdf_text, start_page=None, end_page=None, similarity_threshold=0.5):

    aligned_data = []

    total_pages = len(pdf_text)
    start_index = start_page - 1 if start_page is not None else 0
    end_index = end_page if end_page is not None else total_pages

    selected_pages = pdf_text[start_index:end_index]

    last_matched_page = start_page or 1
    last_matched_line_index = 0

    for audio_segment in audio_transcription:
        best_match = None
        highest_similarity = 0

        for page_dict in selected_pages:
            for page_num, lines in page_dict.items():
                page_num = int(page_num.split('_')[1])

                if page_num < last_matched_page:
                    continue

                for line_index, line_data in enumerate(lines):
                    line_text = line_data['text']

                    similarity = calculate_text_similarity(
                        audio_segment['text'],
                        line_text
                    )
                    context_bonus = 0
                    if page_num == last_matched_page:
                        line_distance = abs(line_index - last_matched_line_index)
                        context_bonus = max(0, 1 - (line_distance * 0.1))

                    adjusted_similarity = similarity + context_bonus

                    if adjusted_similarity > highest_similarity and adjusted_similarity >= similarity_threshold:
                        highest_similarity = adjusted_similarity
                        best_match = {
                            'audio_text': audio_segment['text'],
                            'audio_start': audio_segment['start'],
                            'audio_end': audio_segment['end'],
                            'pdf_text': line_text,
                            'pdf_coords': line_data['coord'],
                            'page_num': page_num,
                            'line_index': line_index,
                        }

        if best_match:
            aligned_data.append(best_match)
            last_matched_page = best_match['page_num']
            last_matched_line_index = best_match['line_index']

    return aligned_data
```

Cet algorithme aligne des segments audio transcrits avec des lignes de texte PDF. Il combine une similarité lexicale floue (fuzz.partial_ratio) avec un bonus de proximité contextuelle (position relative dans la page), ce qui permet de pondérer les erreurs de transcription tout en réduisant les faux positifs.