Outil de Traitement d'Images

Front-end Angular & Back-end FastAPI

Front-end

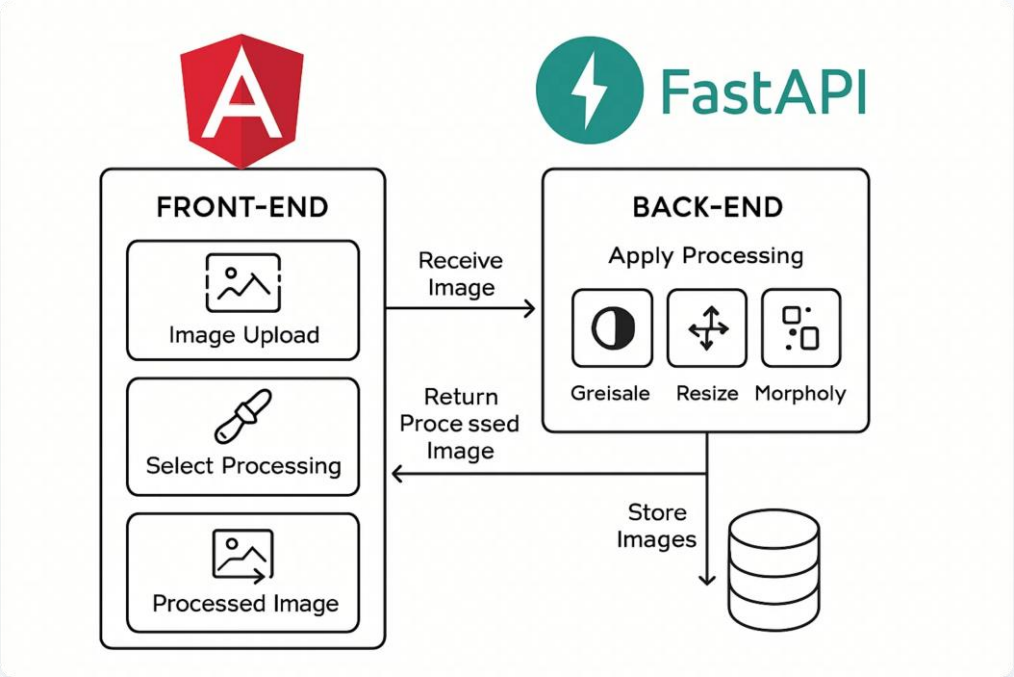
Back-end

Une application web pour le traitement d'images

Téléchargement, traitement et récupération d'images

Juin 2025

Architecture du Projet



Communication

Requêtes HTTP entre le front-end et le back-end pour l'échange d'images et de paramètres

Sécurité

Validation des données et gestion des erreurs pour assurer la robustesse de l'application

Évolutivité

Architecture modulaire permettant d'ajouter facilement de nouveaux traitements d'images

Front-end Angular

Téléchargement d'Images

Composant permettant aux utilisateurs de sélectionner et d'envoyer des images au serveur.

```
<input type="file" (change)="onFileSelected($event)"
      accept="image/*">
```

Sélection des Traitements

Interface permettant de choisir et configurer les traitements à appliquer.

```
<mat-checkbox [(ngModel)]="grayscale">
  Niveaux de gris
</mat-checkbox>
```

Affichage des Résultats

Visualisation de l'image originale et de l'image traitée côte à côte.

```
<div class="image-preview">
  <img [src]="processedImageUrl">
</div>
```

Téléchargement des Résultats

Bouton permettant de télécharger l'image traitée sur l'appareil de l'utilisateur.

```
<a [href]="processedImageUrl" download>
  Télécharger l'image
</a>
```

Back-end FastAPI

Réception des Images

API pour recevoir et valider les images téléchargées depuis le front-end.

```
@app.post("/upload/")
async def upload_image(
    file: UploadFile = File(...)
):
    return {"filename": file.filename}
```

Traitement des Images

Fonctions pour appliquer différents traitements aux images.

```
def process_image(
    image,
    grayscale=False,
    resize=None,
    morphology=None
):
    # Traitement de l'image
    return processed_image
```

Retour des Résultats

API pour renvoyer l'image traitée au format approprié.

```
@app.get("/processed/{image_id}")
async def get_processed_image(
    image_id: str
):
    return FileResponse(
        f"processed/{image_id}.png"
```

Validation et Sécurité

Validation des entrées et gestion des erreurs pour une application robuste.

```
class ImageParams(BaseModel):
    grayscale: bool = False
    resize: Optional[Tuple[int, int]] = None
    morphology: Optional[str] = None
```

Traitements d'Images

Niveaux de Gris

Conversion d'une image couleur en une image en noir et blanc, réduisant chaque pixel à une seule valeur d'intensité.

```
# OpenCV
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# Scikit-image
gray = rgb2gray(img)
```

Redimensionnement

Modification de la taille de l'image tout en préservant ou non ses proportions selon les besoins.

```
# OpenCV
resized = cv2.resize(img, (width, height))

# Scikit-image
resized = resize(img, (height, width))
```

Opérations Morphologiques

Transformations basées sur la forme des objets dans l'image, comme l'érosion, la dilatation, l'ouverture et la fermeture.

```
# OpenCV
kernel = np.ones((5,5), np.uint8)
dilated = cv2.dilate(img, kernel)

# Scikit-image
dilated = dilation(img, square(5))
```

Avantages du Traitement d'Images

OpenCV vs Scikit-image

OpenCV

Bibliothèque de vision par ordinateur optimisée pour la performance

Performance ●●●●●
Facilité d'utilisation ●●●○○
Intégration Python ●●●○○

Scikit-image

Bibliothèque Python pour le traitement d'images scientifiques

Performance ●●●○○
Facilité d'utilisation ●●●●●
Intégration Python ●●●●●

Critère	OpenCV	Scikit-image
Langage principal	C/C++ avec wrappers Python	Python pur
Fonctionnalités	Très nombreuses (2500+ algorithmes)	Moins nombreuses mais bien implémentées
Applications temps réel	Très adapté	Moins adapté
Documentation	Complète mais parfois complexe	Claire et accessible
Cas d'utilisation idéal	Applications nécessitant des performances élevées, traitement vidéo en temps réel	Projets de recherche, prototypage rapide, intégration avec d'autres bibliothèques scientifiques Python

Conclusion

Points Clés du Projet

- Architecture moderne avec séparation claire entre front-end et back-end
- Interface utilisateur intuitive pour le téléchargement et le traitement d'images
- Multiples options de traitement d'images disponibles (niveaux de gris, redimensionnement, opérations morphologiques)
- Possibilité de choisir entre OpenCV et Scikit-image selon les besoins spécifiques

Perspectives Futures

- Ajout de nouveaux filtres et effets (flou gaussien, détection de contours, etc.)
- Intégration de fonctionnalités d'intelligence artificielle pour la reconnaissance d'objets
- Développement d'une version mobile de l'application
- Ajout d'une fonctionnalité de traitement par lots pour plusieurs images

Technologies Utilisées

Merci de votre attention !

Des questions ?