# Mise en place d’un Espace Numérique de Travail à l’EST Salé

# Dispositifs Généraux

L'objectif de ce projet est de mettre en place un **Espace Numérique de Travail (ENT)** pour l'EST Salé basé sur une architecture micro-services modulaire, sécurisée et évolutive. Cet ENT sera augmenté par de l'IA (conversationnelle et générative).

La solution sera basée sur un développement en architecture micro-services, micro-applications déployées dans un cloud privé à base de serveurs Linux (Ubuntu 24.10) hébergés à l'EST Salé et basé sur l'architecture VMware ESXi (hyperviseur de type 1 (Bare Metal)). Afin de faciliter le déploiement et l''évolutivité l'ensemble des micro-services seront conteneurisés et orchestrés.

L'intelligence artificielle qui a été retenue par l'EST Salé pour une utilisation en Cloud Privé est Ollama

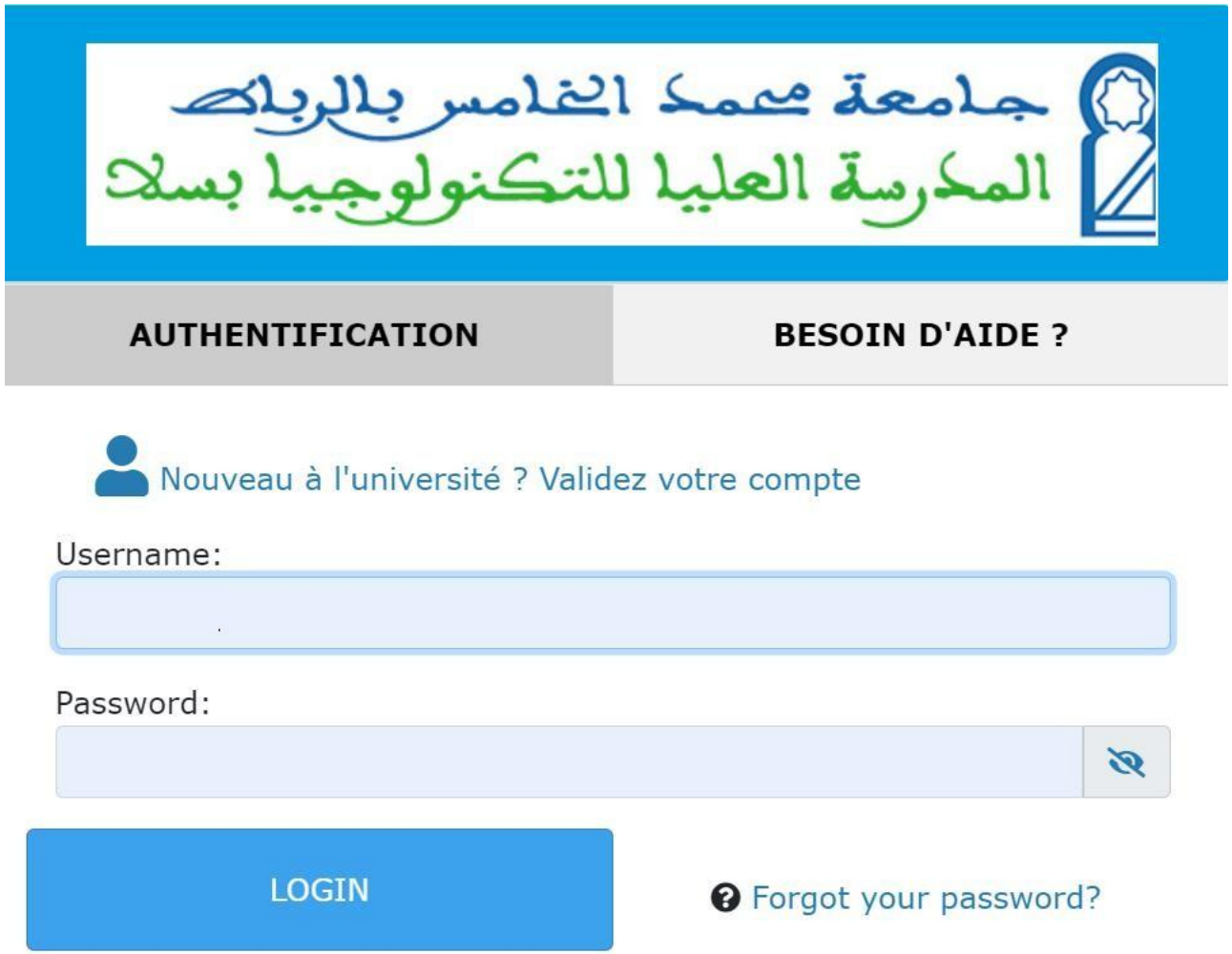
**Ollama**

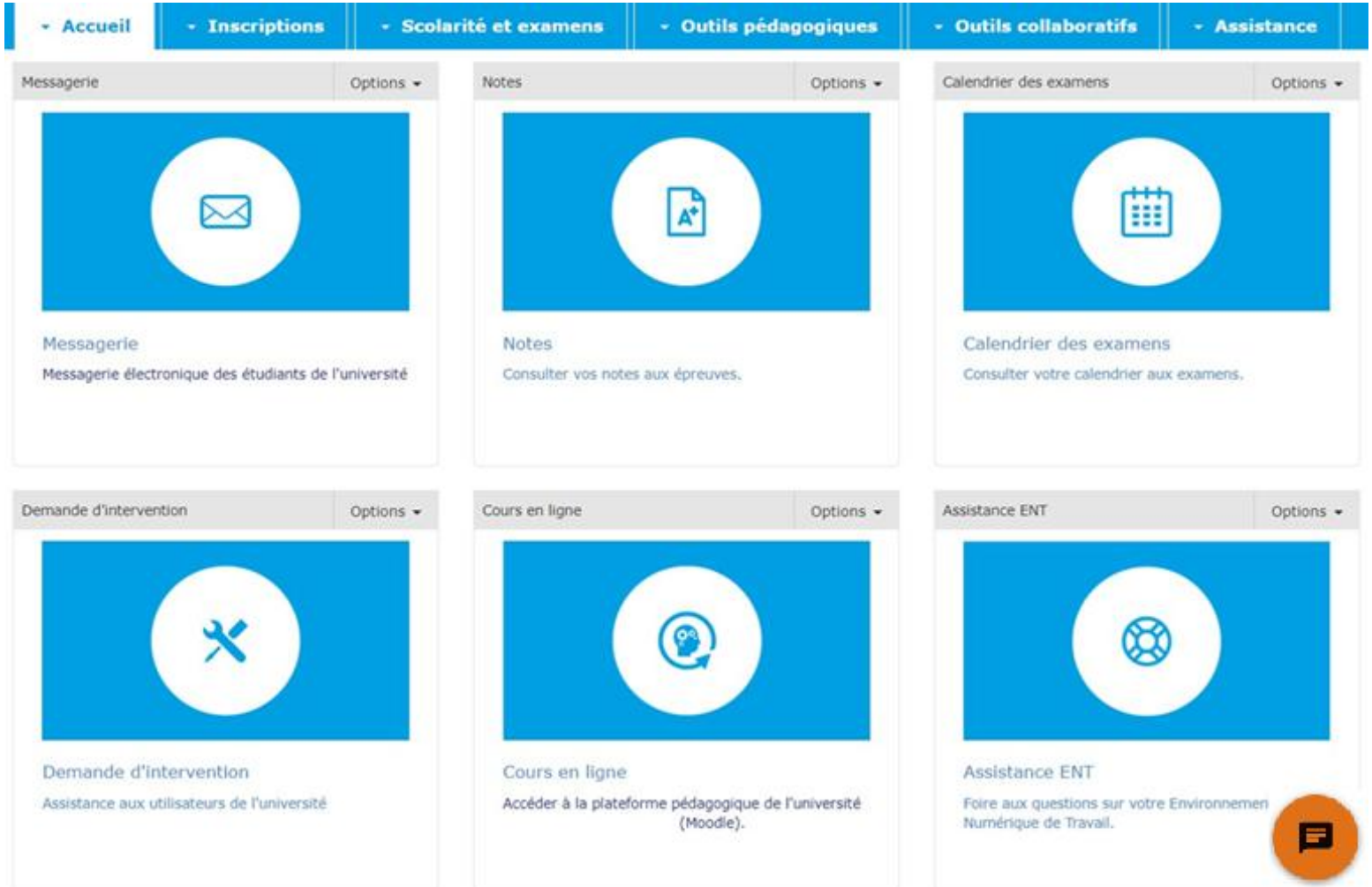
Ollama est un outil puissant permettant d'exécuter localement des modèles de langage de grande taille (LLM) open-source. Compatible avec divers modèles tels que Llama 3 et Code Llama, Ollama simplifie l'utilisation des LLM en regroupant les poids du modèle, la configuration et les données dans un paquet unique, défini par un Modelfile. Cette approche intégrée facilite le déploiement et l'utilisation de LLM sophistiqués sur des machines personnelles.

Llama 3 est la nouvelle génération du grand modèle de langage (LLM) *open source* de Meta. Voici tout ce que vous devez savoir sur ce modèle d'IA qui positionne Meta AI comme l'assistant IA gratuit le plus avancé.

Meta Llama 3 existe en deux principales variantes : Llama 3 8b avec 8 milliards de paramètres et Llama 3 70b avec 70 milliards. Chacune est ensuite déclinée en version Instruct, un modèle affiné pour mieux suivre les instructions des humains. Ces modèles sont donc plus adaptés pour une utilisation en tant que chatbots.

**Espace Numérique de Travail :**




Accueil
Inscriptions
Scolarité et examens
Outils pédagogiques
Outils collaboratifs
Assistance
Messagerie
Options
Messagerie
Messagerie électronique des étudiants de l'université
Notes
Options
Notes
Consulter vos notes aux épreuves.
Calendrier des examens
Options
Calendrier des examens
Consulter votre calendrier aux examens.
Demande d'intervention
Options
Demande d'intervention
Assistance aux utilisateurs de l'université
Cours en ligne
Options
Cours en ligne
Accéder à la plateforme pédagogique de l'université (Moodle).
Assistance ENT
Options
Assistance ENT
Foire aux questions sur votre Environnemen
Numérique de Travail.

Bienvenue

Nous sommes là pour vous aider

Lancer une conversation

Envoyez-nous un message

Que recherchez-vous ?

Rechercher un sujet...

Parcourez cette page avec nous

Rejoindre la co-navigation

L'**Espace Numérique de Travail (ENT)** sera basé sur une architecture micro-services modulaire, sécurisée et évolutive. Cet ENT sera augmenté par de l'IA (conversationnelle et générative) :

**Définition des micro-services**

Chaque service doit être indépendant et répondre à un besoin précis :

- **Service d'authentification** : Gestion des connexions avec OAuth2, JWT ou Keycloak.
- **Service de gestion des utilisateurs** : Inscription, profils étudiants, enseignants, administrateurs.
- **Service de gestion des cours** : Création, modification et consultation des cours en ligne.
- **Service de gestion des fichiers** : Partage et stockage de documents (avec AWS S3, MinIO, ou Google Drive API).
- **Service de messagerie et notifications** : Envoi d'e-mails et notifications push (via RabbitMQ ou Kafka).
- **Service de calendrier et emploi du temps** : Intégration avec Google Calendar ou un système interne.
- **Service de forum et chat** : Communication entre étudiants et enseignants (via WebSockets ou Firebase).
- **Service de gestion des examens et devoirs** : Soumission, correction, et notation en ligne.

**Technologies utilisées :**

- Backend : Python (FastAPI)
- Frontend : React, Angular ou Vue.js pour une interface interactive.
- Base de données : Cassandra
- Stockage de fichiers : MinIO
- Authentification : OAuth2 (Keycloak)
- Déploiement : Ubuntu sur machines virtuelles VMware ESXi
- Communication : API REST
- Conteneurisation et orchestration : Docker + Kubernetes
- Modèle Llama 3 pour l'IA Ollama

**Architecture du Microservice :**

- Cassandra pour stocker les informations et les méta-data des cours, sessions ….
- MinIO pour stocker les fichiers associés aux cours
- Un système d'authentification OAuth2

**Flux de travail :**

- Un enseignant ajoute un cours via l'API (titre, description, fichiers associés).
- Les données sont stockées dans Cassandra et les fichiers dans MinIO.
- Un étudiant peut récupérer la liste des cours et télécharger les fichiers.
- Un étudiant peut interroger le Chat Bot pour obtenir des informations.

**Quelques notions sur le développement d'une application basée sur les micro-services.**

Le développement d'une application basée sur les micro-services repose sur une architecture modulaire où chaque service est indépendant, scalable et interagit avec les autres via des API. Voici les principales étapes pour développer une telle application :

**1. Définir l'architecture et les services**

- Découpage des services : Identifier les fonctionnalités à séparer en micro-services (ex : authentification, gestion des utilisateurs, paiements, notifications, etc.).
- Définition des responsabilités : Chaque microservice doit être autonome, avec une responsabilité unique (principe Single Responsibility).
- Choix des protocoles de communication : REST (HTTP), gRPC, ou message brokers (Kafka, RabbitMQ) pour la communication asynchrone.

**2. Choisir les technologies adapt**ées

- Backend : Spring Boot (Java/Kotlin), Node.js, Django (Python), Go, .NET Core, etc.
- Base de données :
- SQL (PostgreSQL, MySQL) pour les services nécessitant des transactions complexes.
- NoSQL (MongoDB, Cassandra, Redis) pour la scalabilité et les données non relationnelles.
- Message Brokers : Kafka, RabbitMQ ou Redis Pub/Sub pour la communication asynchrone.

- Conteneurisation : Docker pour isoler les services.
- Orchestration : Kubernetes (K8s) ou Docker Swarm pour gérer le déploiement et le scaling des services.

**3. Implémenter chaque microservice**

- Développement indépendant : Chaque service est un projet distinct, pouvant être codé avec une technologie différente.
- Gestion des API :
- Utilisation d'API Gateway (ex : Kong, API Gateway AWS, Nginx, Traefik) pour gérer les requêtes.
- Swagger/OpenAPI pour la documentation des API.
- Sécurisation : JWT, OAuth2, Keycloak pour la gestion des authentifications et autorisations.

**4. Gérer la communication entre microservices**

- Synchronisation : API REST ou gRPC pour des appels directs entre services.
- Asynchronisation : Message Brokers (Kafka, RabbitMQ) pour réduire la dépendance entre services.
- Service Discovery : Consul, Eureka (Spring Cloud), ou Kubernetes Service pour détecter dynamiquement les services.

**5. Déploiement et monitoring**

- CI/CD : GitHub Actions, GitLab CI/CD, Jenkins pour automatiser le build et le déploiement.
- Observabilité :
- Logging : ELK Stack (Elasticsearch, Logstash, Kibana), Fluentd.
- Monitoring : Prometheus, Grafana, Datadog.
- Tracing : OpenTelemetry, Jaeger.

**6. Tester et optimiser**

- Tests unitaires et d'intégration : JUnit, Mocha, Jest.
- Tests de charge : JMeter, k6.
- Tests end-to-end : Cypress, Selenium.
- Optimisation : Load balancing, scaling automatique, cache (Redis, Memcached).

**7. Déploiement en production**

- Choix du cloud : AWS (EKS, ECS, Lambda), Google Cloud (GKE), Azure (AKS) ou solutions on-premise.
- Gestion des erreurs et résilience : Circuit Breaker (Hystrix, Resilience4j).
- Mise à jour continue : Canary Deployments, Blue-Green Deployments.

**Architecture des microservices Projet**

**Microservice 1 : Core (Auth)**

- Gère l'authentification des utilisateurs.
- Fournit un token JWT après connexion via Keycloak.

**Microservice 2 : Ajout de Fichiers**

- Permet aux enseignants d'ajouter des cours.
- Stocke les fichiers dans MinIO.
- Stocke les métadonnées (titre, description, URL) dans Cassandra.
- Nécessite une authentification via un token JWT.

**Microservice 3 : Téléchargement**

- Permet aux étudiants de lister et télécharger les cours disponibles.
- Récupère les métadonnées depuis Cassandra.
- Fournit un lien sécurisé pour télécharger les fichiers depuis MinIO.
- Nécessite une authentification via un token JWT.

**Microservice 4 : Administration**

- Permet aux administrateurs de créer les utilisateurs et les rôles.
- Récupère les métadonnées depuis Cassandra.
- Nécessite une authentification via un token JWT.