



Intelligent Translation of Clinical Documents with AI - TCC

Project team members

Full Name
Abner Silva Barbosa

Introduction

This report presents a detailed summary of the activities developed throughout the first module in the Entrepreneurship track. Our goal was to initiate the development of an innovative solution that uses advanced Artificial Intelligence to translate clinical and pharmaceutical technical documents, maintaining specialized terminology, original formatting, and precision necessary for the regulatory and research context. The solution aims not only to translate text but to preserve complex elements such as tables, graphs, and images with captions, ensuring the integrity of technical documents.

1. Module Goals

The primary objective of this module was to implement optimizations in the initial architecture, leveraging technologies, tools, services, and other key factors, building upon the foundation established in the previous module. Additionally, a secondary yet crucial goal was to develop a comprehensive pricing strategy for our solution and business model, ensuring preparedness for sales and investor engagement.

Key Outcomes:

1. **Optimization of Architecture:** Enhance the existing architecture by integrating optimized technologies, tools, and services to improve performance, scalability, and efficiency.
2. **Business Modeling and Pricing Strategy:** Develop a robust business model and pricing strategy to establish a solid foundation for sales, investor relations, and revenue growth.
3. **Cost Analysis and Margin Identification:** Determine the minimum costs associated with our application to inform margin calculations and ensure financial sustainability.
4. **Integration with Frontend and Backend:** Achieve seamless integration between frontend and backend components to ensure a cohesive and efficient user experience.
5. **LLM (Large Language Model) Enhancement:** Refine the LLM generated with contextualized document data to improve its accuracy, relevance, and overall performance.

By achieving these objectives, we aim to establish a robust, scalable, and commercially viable solution that is well-positioned for success in the market.

2. Technical Advancements

This section highlights the significant technical advancements and improvements made to the project, encompassing changes in technologies, infrastructure, tool additions, service enhancements, code optimizations, architectural improvements, and security enhancements.

2.1 Transition to Own Infrastructure

- Initially, we considered using Supabase as our backend solution. However, after evaluating the costs and requirements of our project, we decided to replace it with MinIO. While Supabase is a robust tool, our Minimum Viable Product (MVP) needs led us to prioritize a more cost-effective and controllable solution.
- After switching to MinIO as our object storage solution, we proceeded with a full configuration to ensure performance, scalability, and security. We set up MinIO within our own infrastructure, enabling us to have complete control over file storage and access policies. Encryption at rest was implemented to protect sensitive clinical documents, aligning with data protection regulations such as LGPD and GDPR. This setup also allowed us to integrate MinIO seamlessly with our translation pipeline and document processing service, reducing latency and improving data flow across components.

2.2 Backend Enhancements

- To improve database reliability and developer productivity, we integrated **Prisma** as our ORM (Object-Relational Mapping) tool. Prisma allowed us to

define our database schema with clarity and apply version-controlled migrations seamlessly. This transition improved our ability to manage relationships between multi-tenant entities and user permissions, making the codebase more maintainable and reducing the likelihood of runtime errors. Additionally, Prisma's type-safe queries aligned well with our TypeScript-based backend, improving consistency across services.

- As part of our architectural improvements, we implemented **asynchronous task queues using RabbitMQ**. This allowed us to decouple document processing from the user request-response cycle, significantly improving system responsiveness and scalability. Uploaded documents are now placed in a queue, enabling background workers to handle time-consuming operations such as text extraction, format preservation, and translation without blocking the frontend. This also opened the door for better monitoring, retry mechanisms, and future horizontal scaling of workers.
- We conducted a complete refactor of the API endpoints related to document upload and translation to improve maintainability and performance. The new flow separates responsibilities more clearly across services and includes improved validation, error handling, and logging. The translation API was redesigned to support multi-tenant access, detect the document's base language, and integrate with the asynchronous processing pipeline. This restructuring also laid the groundwork for new features such as translation progress tracking and user-specific document history.

2.3 Multi-Tenant Architecture

- To support a scalable multi-tenant model, we restructured our database schema to isolate data between organizations while maintaining operational efficiency. We introduced a `tenant_id` field across relevant tables, ensuring

that each user and document is associated with a specific organization. This change laid the foundation for tenant-aware queries, allowing us to manage multiple clients securely within the same infrastructure without data leakage or overlap.

- As part of our security strategy, we studied and began implementing **Row-Level Security (RLS)** to enforce strict data isolation at the database level. With RLS, PostgreSQL restricts access to rows based on tenant context, adding an additional protection layer beyond application-level logic. This approach ensures that users from Tenant A cannot access any data from Tenant B, even in edge cases or potential misconfigurations. RLS aligns with our commitment to privacy and regulatory compliance in handling sensitive clinical information.
- To validate our multi-tenant approach before full deployment, we designed a simplified version of the system architecture. This lean version included key components such as authentication, document processing, and storage, all scoped to simulate isolated tenant behavior. It allowed us to test concepts like dynamic routing by tenant, access control, and performance implications without the overhead of a complete production environment. Insights gained from this prototype helped guide the full architecture evolution with greater confidence.

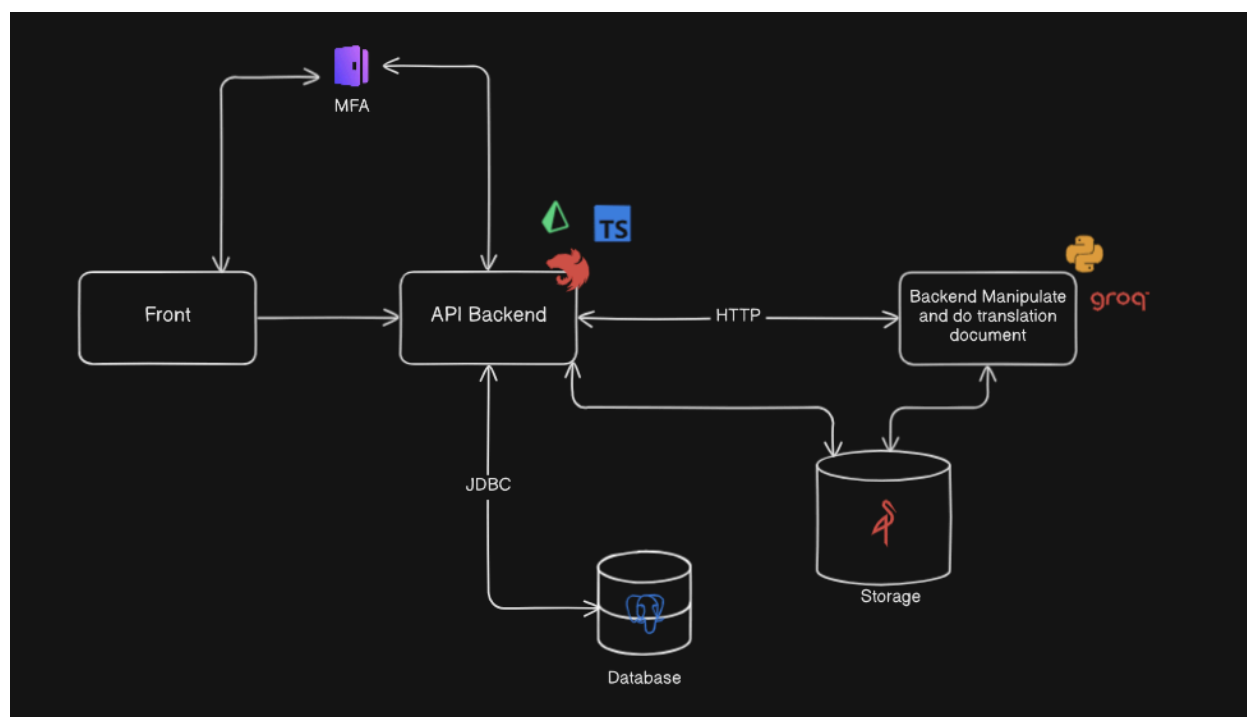
2.4 Authentication & Authorization

- To strengthen authentication and user access control, we implemented **Logto**, an open-source identity provider supporting OAuth 2.0, RBAC (Role-Based Access Control), and MFA (Multi-Factor Authentication). Logto was chosen for its modern developer experience, integration flexibility, and native support for multi-tenant applications. We configured role hierarchies

for different permission levels (e.g., admin, reviewer, contributor) and enabled optional MFA to increase security, especially for users handling sensitive documents in regulated environments.

- We established full integration between the frontend and backend services using Logto's SDKs and APIs. On the frontend, we implemented secure login flows with token handling and tenant-aware routing. On the backend, authentication middleware was added to validate access tokens and enforce role-based permissions at the API level. This setup ensures seamless and secure user sessions across the entire platform, while also laying the groundwork for audit logging and future access analytics.

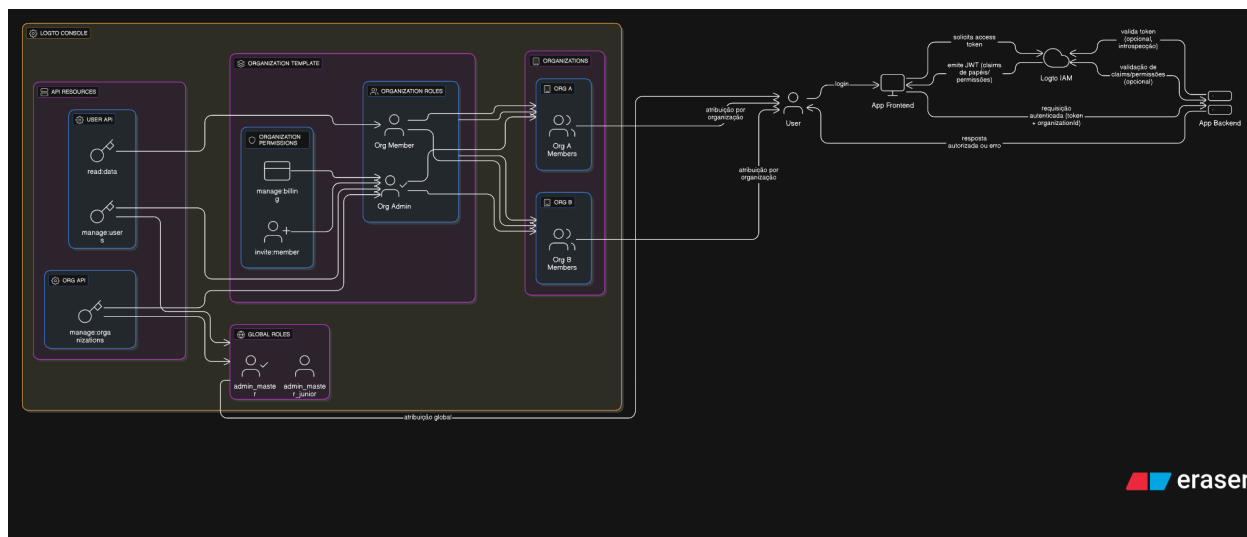
Prototype Architecture:



The prototype architecture follows a modular and scalable design focused on secure document processing and translation. The **Frontend** interacts with the **API Backend**, which is developed in **TypeScript** and handles authentication via an external **MFA provider (Logto)**. Upon authentication, users can upload clinical documents, which

are stored securely using an object **Storage** system (MinIO). The **API Backend** communicates via **HTTP** with a specialized **Python-based backend service**, responsible for manipulating and translating the documents using **Groq's LLMs**. The system maintains data persistence through a **relational database**, accessed via **JDBC**, supporting multi-tenant logic and user-role associations. This architecture prioritizes security, modularity, and performance, while remaining cost-effective for the MVP stage.

Logto strategy:



The system leverages **Logto IAM** to manage authentication and authorization securely, using **OAuth 2.0**, **MFA**, and a robust **Role-Based Access Control Architecture (RBAC)**. Each user is assigned roles and permissions at both the **global** and **organization** levels. Organizations (e.g., Clinic A or B) are isolated, with specific roles such as **Org Admin** and **Org Member** controlling access to features like billing, user invitations, and document management. When a user logs in through the frontend, Logto issues a **JWT token** containing claims about their roles and permissions. This token is then validated by the backend, enabling secure and scoped access to resources across multiple tenants. The architecture supports scalability and maintains strict separation of concerns between organizations.

Data isolation strategy:



There are several architectural strategies to implement multi-tenancy in software systems. The first approach involves separate databases per tenant, managed by a master controller – offering strong isolation but with higher complexity and cost. Other models include schema-per-tenant or schema-per-table strategies, each with different trade-offs in scalability and maintenance.

For this project, we adopted the **"Shared Database with Tenant Identifier"** model (Option 2 in the diagram). In this approach, all tenants share the same database, but each row in the main tables (e.g., users, documents) includes a **Tenant ID** to separate data logically. This model is cost-effective, simplifies infrastructure management, and supports the use of **Row-Level Security (RLS)** for secure data isolation between organizations, making it a practical and scalable solution for our MVP stage.

3. AI and Translation Optimization

In this module, a key focus was placed on optimizing the use of Artificial Intelligence to deliver high-quality clinical translations with reduced operational costs. As we move from experimentation to building a scalable MVP, we explored alternatives to commercial LLMs, refined our translation pipeline, and addressed technical challenges related to alignment and layout preservation. This section details the evolution of our AI stack, including the migration to cost-effective models, improvements in text span mapping, and strategies to maintain document structure during the translation process. These efforts were crucial to make the solution viable for real-world clinical use, where accuracy, performance, and cost-efficiency must coexist.

3.1 Migration to Cost-Effective Models

As part of our cost optimization strategy, we transitioned from using OpenAI models to **Gemini**, and ultimately to [Groq](#)—leveraging models such as **Meta’s Mistral and Maverick 70B**. Groq provided significant advantages in speed and cost, offering high-quality translation outputs without the infrastructure expenses associated with self-hosting large models. This migration allowed us to scale the translation process more sustainably during MVP development, while maintaining acceptable levels of technical precision.

3.2 Span Mapping and Layout Preservation

A critical challenge in document translation is maintaining alignment between the original and translated content, especially in richly formatted PDFs. To address this, we developed initial algorithms to **map spans of text between source and target**

versions, ensuring positional consistency. Additionally, we implemented a fallback mechanism for **missing fonts**, where the system detects absent fonts in documents and substitutes them with visually similar ones, preserving the document's appearance and readability.

4. Document Processing and Rendering

To improve flexibility in document manipulation, we conducted tests converting PDFs into **HTML and DOCX formats**, which showed greater promise for editability and layout control. These formats facilitated downstream tasks such as preserving tables and embedded elements. Alongside this, we researched and integrated **scripts to automatically download required fonts**, reducing rendering issues. On the frontend, we added a **React-based PDF viewer**, allowing users to interactively review translated documents with real-time visual fidelity.

5. Frontend Evolution

The user interface underwent significant development during this module. We designed and implemented key screens such as **login, document upload, billing management, user administration, and document review**. Navigation was enhanced with a responsive sidebar, and smooth **page animations** were introduced to improve the user experience. Additionally, we focused on **real-time rendering** of PDF documents, enabling users to view original and translated content side-by-side within an accessible and collaborative interface.

[Click here for see the frontend deployed](#) (Mocked)

user: admin@gmail.com

pass: 123123123

Aqui você pode verificar os documentos e as suas traduções



Todos Em progresso Completos

 Pesquise por nome do documento...

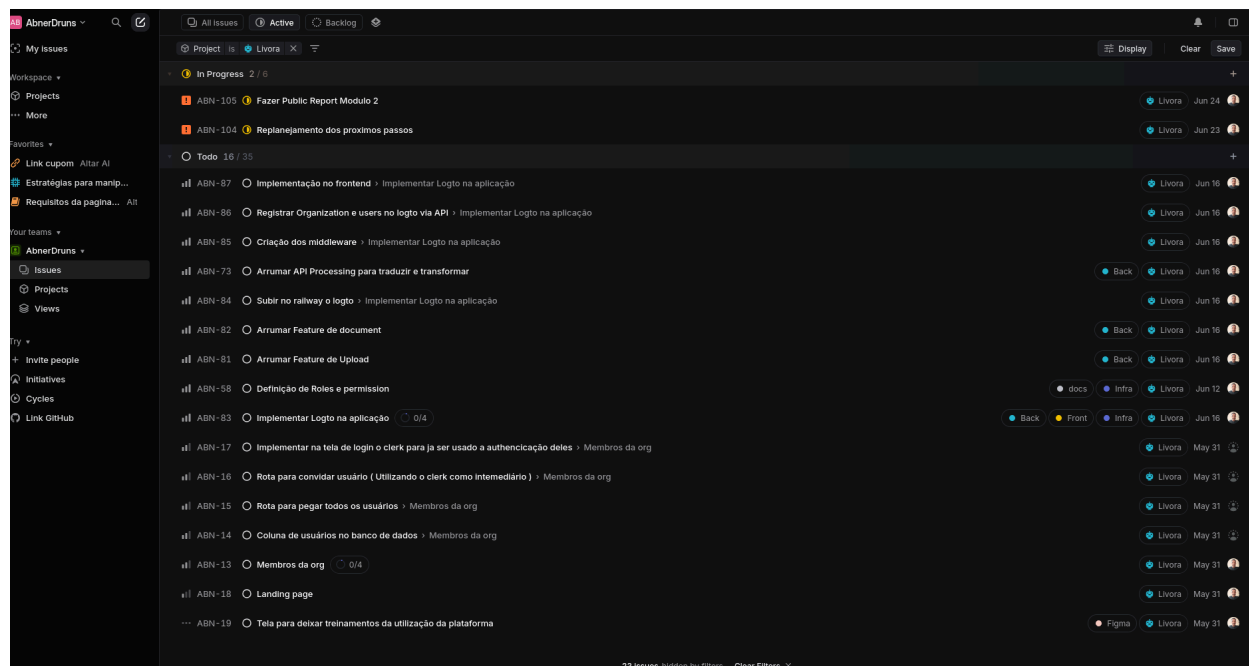
Nova tradução

Filtro

6 documentos

Nome do documento	Data de upload	Revisor	Status	
 ICF_003.pdf	13/05/2025	GB Gabriel Braga	Em progresso	⋮
 ICF_003.pdf	13/05/2025	GB Gabriel Braga	Concluído	⋮
 ICF_003.pdf	13/05/2025	GB Gabriel Braga	Erro	⋮
 ICF_003.pdf	13/05/2025	GB Gabriel Braga	Em revisão	⋮
 ICF_003.pdf	13/05/2025	GB Gabriel Braga	Em revisão	⋮
 ICF_003.pdf	13/05/2025	GB Gabriel Braga	Concluído	⋮

1 2 3 4 5 6 7 8



7. Business and Market Validation

We began exploring business models through insights gathered in **Growth Hacking classes**, applying frameworks such as funnels and performance metrics to the project. Parallel to this, we conducted strategic discussions on **pricing models and positioning for B2B clients**, particularly mid-sized clinical research organizations. Meetings with key stakeholders—including **IDOR representatives, healthcare professionals, and fellow Inteli students**—offered actionable insights into the market's real needs and regulatory demands.

8. Research Highlights

Throughout the module, we investigated **low-resource LLM alternatives**, including **LLMs in BITS**, to evaluate the feasibility of running lightweight models in production. We also initiated a **patent search** and brand analysis process for our platform name,

Livora, exploring legal pathways for protection. Additionally, we studied **security certifications** and compliance standards relevant to clinical systems, laying the groundwork for future alignment with HIPAA, LGPD, and similar frameworks.

<https://github.com/microsoft/BitNet>

9. Challenges Faced

One of the major ongoing challenges has been **maintaining visual layout integrity during translation**, especially in documents containing dense formatting, charts, and tables. Our span mapping strategy is still evolving and presented **technical limitations** in accurately aligning translated text. Furthermore, achieving high translation quality while **balancing performance and cost constraints** remains a delicate trade-off—particularly when considering model latency, formatting fidelity, and the sensitivity of clinical content.

Research link:

- <https://docs.google.com/document/d/1G0H7HvrbPf3VGxqGw1VieeSGYL8qt2IJ/edit?usp=sharing&oid=102944795941725922945&rtpof=true&sd=true>
- <https://g.co/gemini/share/76bc61325391>

10. Next Steps

- Finalize multi-tenant implementation and test scaling scenarios.
- Improve PDF conversion and translation alignment.
- Expand real-world document testing with clinical partners.
- Strengthen user access management and security layers.

11. Deliverables – Module 2 Summary

Sprint 1 (April 22 – May 3)

- Document structuring for real clinical context.
- Migration from Supabase to MinIO to reduce cost and gain infra control.
- Initial implementation of logging, landing page (V0.dev), and UI corrections.
- Deep dive into **LLM cost optimization** and font handling strategies.
First discussions about switching to own infra, downloading fonts, and creating fallback mechanisms.

Sprint 2 (May 6 – May 17)

- Frontend: login screen, sidebar, navigation, document review and animation.
- Backend: base implementation of file upload and language detection.
- UX improvements in Figma, creation of payment and user management screens.
- Research on PDF rendering libraries and lightweight translation models.
- Early integration with **Groq** and improvements in translation pipeline.

Sprint 3 (May 20 – May 31)

- Implementation of the new EN → PT_BR model using Groq.
- Optimizations in Prisma + database architecture for multi-tenant logic.
- Study of conversion from PDF to DOCX/HTML for better editability.
- Real-time collaboration tests and backend refactor to support new flows.
- Security studies: RBAC, MFA, OAuth2, and Row-Level Security (RLS) for tenants.

Sprint 4 (June 3 – June 14)

- Meetings with mentors (Pedro, Vinícius, Cláudio from IDOR).
- Study and validation of multi-tenant strategy using simplified architecture.
- Full configuration and partial integration of **Logto** for auth management.
- Further research on PDF conversion flows and alternate LLM usage.
- Implementation of fallback fonts and alignment algorithm experiments.

12. Key Learnings – Technical and Business Perspectives

Throughout this module, I encountered a wide range of technical and strategic challenges that provided valuable learning opportunities. These experiences contributed significantly to my development both as a software engineer and as a product builder in the healthtech space.

Technical Learnings

- **Architecture Decisions Have Trade-offs:** Migrating from managed services (like Supabase) to self-hosted infrastructure (using MinIO and Groq) deepened my understanding of the trade-offs between control, cost, and complexity. This shift required me to manage more layers of the stack but also gave me performance gains and budget flexibility.
- **Importance of Scalable Design:** Implementing multi-tenant logic using shared database models with tenant-aware queries and Row-Level Security (RLS) pushed me to design a data model that is both secure and scalable. I learned how to structure queries defensively and design APIs that respect tenant boundaries.
- **Asynchronous Workflows Matter:** Introducing RabbitMQ to manage document processing pipelines showed me the real impact of decoupled services. It not only improved performance but made the backend more

resilient and easier to monitor.

- **Secure Auth is Complex but Critical:** Working with Logto exposed me to real-world identity and access management patterns, such as OAuth2, RBAC, and MFA. Understanding token flow, role hierarchies, and secure frontend/backend integration was a major step in building enterprise-grade apps.
- **LLM Integration Requires Strategy:** Switching from OpenAI to Groq using models like Mistral taught me that LLMs are not plug-and-play – optimization involves balancing performance, cost, and output quality, especially for domain-specific language like clinical data.

Business and Product Learnings

- **MVP Scope Needs Ruthless Prioritization:** Trying to deliver too many features early almost led to technical overload. This taught me to focus on the core flow (document upload → translation → review) and deprioritize less critical features.
- **Customer Feedback Shapes Features:** Conversations with mentors, clinicians, and regulatory stakeholders revealed needs we hadn't fully considered (e.g., font preservation and audit logs). This highlighted the importance of validation before scaling.
- **B2B Pricing Is Complex:** Developing a pricing model made me consider variable costs like API usage, LLM inference time, and support, along with market expectations and perceived value. It became clear that pricing in healthtech requires a mix of cost-based, value-based, and competitor-informed strategies.

- **Security and Compliance Are Not Optional:** Working with clinical content meant understanding data privacy regulations (LGPD, GDPR, HIPAA). Even in an MVP, ignoring these elements would reduce our credibility and viability in the market.