Development Log - Kilometros de Vida

Álvaro - Backend Setup

- Set up initial Express server and MongoDB Atlas connection
- Created basic API endpoints for givers and drivers
- Configured environment variables and `.env.template`
- **Challenge**: MongoDB connection kept timing out. Had to whitelist IP address in Atlas and update connection string format.
- **AI Assist**: Used ChatGPT to debug connection string format - prompt: "My MongoDB Atlas connection is timing out with this error: [error]. Here's my connection string: [string]. What am I doing wrong?"
- **Status**: Backend running on localhost:5050

Regina - Design & Planning

- Created wireframes for main pages (Home, Giver form, Driver form, Profile)
- Chose color palette and typography (blue/teal with serif headings)
- Set up Figma mockups for team review
- Started researching React animation libraries
- **Status**: Design system established

Daniela - Project Structure

- Initialized Vite + React project in `/client` folder
- Set up Tailwind CSS configuration
- Created basic folder structure (components, pages, context)
- **Challenge**: Tailwind wasn't loading styles initially - fixed by updating `tailwind.config.js` paths
- **AI Assist**: Asked Claude: "Tailwind CSS not working in my Vite React project. Styles aren't applying. Here's my config file [code]. What's missing?"
- **Status**: Frontend skeleton ready

Regina - UI Components

- Built Navbar component with responsive menu
- Created Home page with hero section and stats counters
- Implemented Framer Motion animations for fade-ins
- **Challenge**: Mobile menu wasn't closing after clicking links. Added state management to fix it.

- **AI Assist**: Used ChatGPT to generate animated counter component - prompt: “Create a React component that animates numbers counting up from 0 to a target value using Framer Motion”
- **Status**: Landing page looking good

Daniela - Database Schema

- Designed and implemented Users, Givers, and Drivers collections
- Added validation rules for phone numbers and emails
- Created seed script with sample data for testing
- Tested CRUD operations with Postman
- **Status**: Database fully functional

Álvaro - Google OAuth Integration

- Set up Google Cloud Console project
- Implemented OAuth 2.0 on frontend using Google Identity Services
- Created UserContext for managing auth state
- **Challenge**: CORS errors when calling backend. Had to configure CORS middleware properly.
- **AI Assist**: Asked Claude: “Getting CORS error when React app calls Express backend. Frontend on localhost:5173, backend on localhost:5050. How do I fix this?”
- **Status**: Users can log in with Google

Daniela - Forms & Validation

- Built Giver registration form with all required fields
- Added Driver volunteer form
- Implemented form validation (email, phone format)
- Connected forms to backend API
- **Challenge**: Form submissions weren’t clearing after success. Added reset functionality.
- **Status**: Both forms working and saving to database

Álvaro - API Enhancements

- Added `/api/my-donations` endpoint for user-specific data
- Implemented PUT and DELETE routes for donations
- Added email-based authorization checks
- **Challenge**: Delete operation wasn’t returning proper response codes. Fixed status codes and error handling.
- **AI Assist**: Used GitHub Copilot to autocomplete the PUT and DELETE route handlers based on the POST pattern I’d already written
- **Status**: Full CRUD operations complete

Regina - Profile Page

- Created Profile dashboard to display user's donations
- Added Edit/Delete buttons for each donation
- Styled donation cards with hover effects
- **Status**: Users can manage their donations

Regina & Daniela - Map Integration (Pair Programming)

- Researched free geocoding options (Google Maps vs OpenStreetMap)
- Decided on Nominatim API to avoid costs
- Installed React Leaflet for map display
- **Challenge**: Google Maps API pricing too high for student project
- **Status**: Switched to free alternative

Álvaro - Backend Fixes

- Fixed bug where donations without userId were causing errors
- Added better error messages for failed API calls
- Implemented request logging for debugging
- Tested all endpoints thoroughly
- **AI Assist**: Asked ChatGPT: "Write me a simple Express middleware for logging all incoming requests with method, path, and timestamp"
- **Status**: Backend stable and error-free

Daniela - Geocoding Implementation

- Implemented Nominatim geocoding on DonationMap component
- Added loading states while geocoding addresses
- Implemented localStorage caching to reduce API calls
- **Challenge**: Rate limiting (1 req/sec) was slowing down map. Added delays between requests.
- **AI Assist**: Asked Claude: "How do I add a delay between API calls in JavaScript to respect a rate limit of 1 request per second?"
- **Status**: Map showing all donation locations

Regina - Polish & Animations

- Added Framer Motion animations throughout the site
- Implemented smooth page transitions
- Created loading spinners for async operations
- Refined responsive design for mobile devices
- **Status**: UI feeling smooth and professional

Daniela - Edit Functionality

- Built edit modal for updating donations
- Pre-populated form with existing donation data
- Connected edit form to PUT endpoint
- **Challenge**: Modal wasn't closing after successful update. Added state reset.
- **AI Assist**: Used GitHub Copilot to suggest the modal component structure - it autocompleted most of the JSX after I wrote the opening tags
- **Status**: Edit feature complete

Álvaro - Deployment Attempt #1

- Tried deploying backend to Render
- Frontend deployed to Netlify
- **Challenge**: Render kept crashing with memory errors and was extremely slow
- **Status**: Looking for alternative hosting

Daniela - Deployment Fix

- Researched alternative hosting platforms
- Switched frontend from Netlify to Vercel
- Updated environment variables for production
- **Challenge**: Environment variables weren't loading on Vercel. Had to configure them in dashboard.
- **AI Assist**: Asked ChatGPT: "How do I set environment variables in Vercel for a Vite React app? My VITE_ variables aren't being recognized"
- **Status**: Frontend live on Vercel

Álvaro - Backend Deployment

- Kept backend on Render but optimized memory usage
- Added health check endpoint for monitoring
- Updated CORS to allow Vercel domain
- **Status**: Backend finally stable on Render

Regina - Final Design Touches

- Updated footer with team credits
- Added accessibility improvements (ARIA labels, alt text)
- Fixed contrast issues for better readability
- Tested on multiple browsers (Chrome, Firefox, Safari)
- **AI Assist**: Asked Claude: "Review my React component for accessibility issues - here's the code [component]. What ARIA labels am I missing?"
- **Status**: Design polished and accessible

Final Team Review

- Tested full user flow: Sign up → Create donation → View on map → Edit → Delete
- Verified mobile responsiveness
- Checked that all links work
- **Status**: Project complete and deployed!

Summary of Technical Challenges

Authentication

- **Problem**: Needed user authentication without building complex backend
- **Solution**: Used Google OAuth 2.0 for secure, simple login

Map Visualization

- **Problem**: Google Maps API too expensive for student project
- **Solution**: Switched to free Nominatim geocoding + React Leaflet

State Management

- **Problem**: Sharing user data across components
- **Solution**: Created React Context for global state management

Deployment

- **Problem**: Render was slow and kept crashing
- **Solution**: Moved frontend to Vercel

Form Handling

- **Problem**: Forms not resetting after submission
- **Solution**: Added proper state cleanup on successful submit

CORS Issues

- **Problem**: Frontend couldn't call backend API
- **Solution**: Configured CORS middleware with proper origins

Rate Limiting

- **Problem**: Nominatim API limited to 1 request/second
- **Solution**: Added delays and localStorage caching

What We Learned

Technical Skills

- Full-stack MERN development from scratch
- OAuth 2.0 authentication flow
- API design and RESTful principles
- Geocoding and map integration
- Responsive web design
- Deployment and DevOps basics

What Worked Well

- Dividing work by expertise (backend/frontend/design)
- Using free tiers of services to avoid costs
- Frequent communication via group chat

Impact & Future Vision

This project demonstrates a real solution to food waste in our community. With proper adoption, it could connect dozens of restaurants with volunteer drivers to feed families in need.