DETRAN-SP Citizen Services

Technical Design Document

v2.0 | December 2024

Overview

Agentic chat system for Brazilian driver license renewal (CNH) with Lightning Network payments.

Architecture

Component	Role
UI	React + CopilotKit. Chat interface, frontend tools, forms, QR display.
Agent	Google ADK LlmAgent via ag_ui_adk. Intent detection, orchestration.
API	FastAPI. Write operations: auth, create_ticket, confirm_payment. Integrates WDK Spark.
MCP Server	MCP wrapper around API. Read-only: list_tickets, get_ticket, list_operations.
WDK Spark	Tether wallet SDK. Lightning invoice creation and payment verification.
DB	PostgreSQL. Operations, tickets, users.

Data Flow

- **Reads**: Agent → MCP Server → DB
- **Writes**: UI → API → DB + WDK Spark
- **Frontend Tools**: UI advertises to Agent via CopilotKit

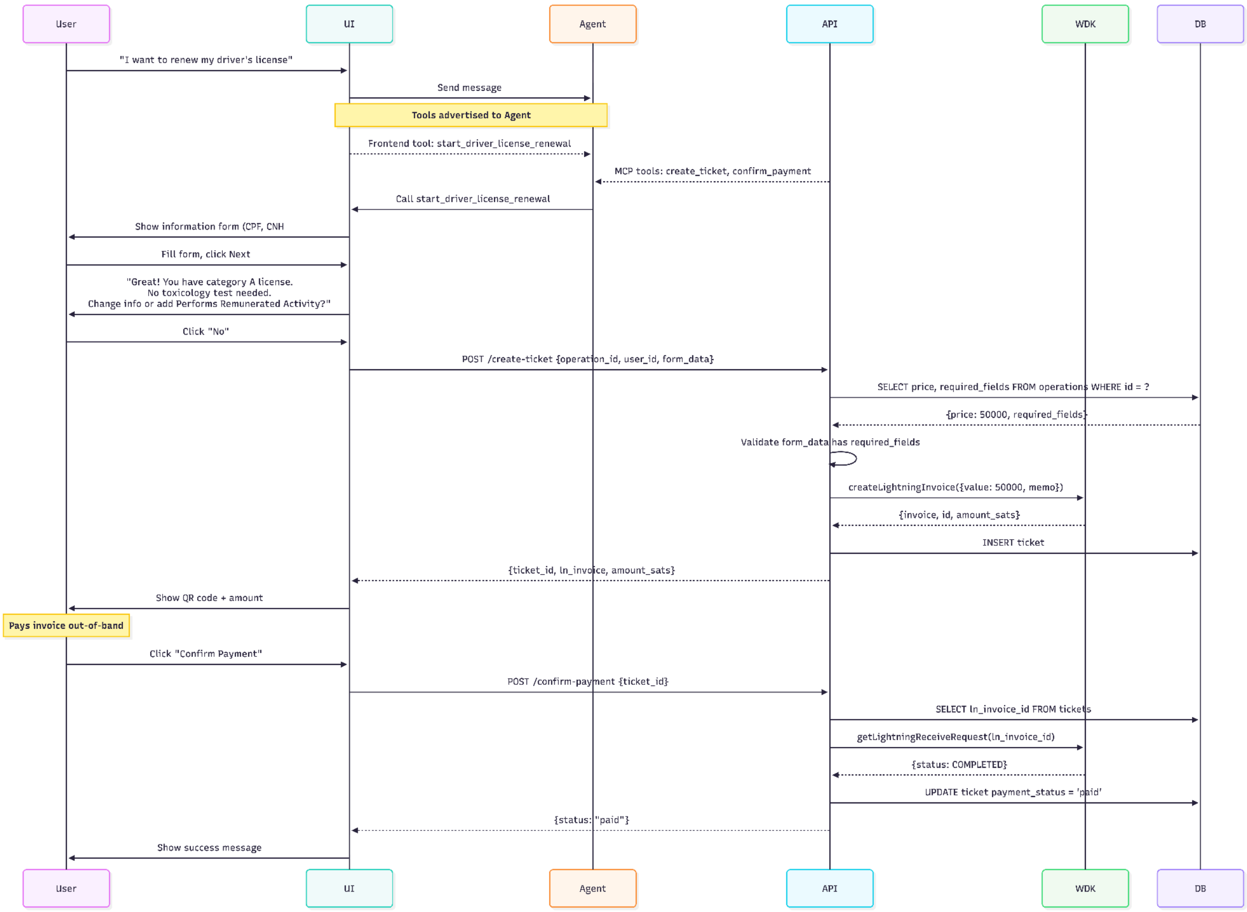
User Flow

1. **Login** — User enters email + password. UI calls POST /auth/login. Receives JWT.
2. **Intent** — User: "I want to renew my driver's license". Agent identifies intent.
3. **Form** — Agent calls start_driver_license_renewal frontend tool. UI shows form: CPF, CNH Number, CNH Mirror.
4. **Confirm Info** — UI shows: "Category A license. No toxicology test needed. Change info or add Performs Remunerated Activity?" User clicks No.
5. **Create Ticket** — UI calls POST /tickets with operation_id and form_data. API validates, creates LN invoice via WDK Spark, inserts ticket, returns invoice.
6. **Payment** — UI displays QR code + amount. User pays out-of-band.
7. **Confirm** — User clicks Confirm Payment. UI calls POST /tickets/:id/confirm-payment. API checks WDK Spark, updates DB, returns status.
8. **Success** — UI shows success message.

Query Flow (via Agent)

User: "What's the status of my ticket #abc123?"
Agent → MCP get_ticket tool → Returns ticket details including payment_status.

Sequence Diagram


User
UI
Agent
API
WDK
DB
"I want to renew my driver's license"
Send message
Tools advertised to Agent
Frontend tool: start_driver_license_renewal
MCP tools: create_ticket, confirm_payment
Call start_driver_license_renewal
Show information form (CPF, CNH
Fill form, click Next
"Great! You have category A license.
No toxicology test needed.
Change info or add Performs Remunerated Activity?"
Click "No"
POST /create-ticket {operation_id, user_id, form_data}
SELECT price, required_fields FROM operations WHERE id = ?
{price: 50000, required_fields}
Validate form_data has required_fields
createLightningInvoice({value: 50000, memo})
{invoice, id, amount_sats}
INSERT ticket
{ticket_id, ln_invoice, amount_sats}
Show QR code + amount
Pays invoice out-of-band
Click "Confirm Payment"
POST /confirm-payment {ticket_id}
SELECT ln_invoice_id FROM tickets
getLightningReceiveRequest(ln_invoice_id)
{status: COMPLETED}
UPDATE ticket payment_status = 'paid'
{status: "paid"}
Show success message
User
UI
Agent
API
WDK
DB

Database Schema

operations

Column	Type	Description
id	SERIAL	Primary key
name	TEXT	Unique identifier (e.g., driver_license_renewal)
description	TEXT	Human-friendly step-by-step explanation
price	INTEGER	Fee in satoshis
required_fields	JSONB	Schema: {field_name: type} for validation

tickets

Column	Type	Description
id	UUID	Primary key
operation_id	INTEGER	FK → operations.id
user_id	TEXT	User identifier
form_data	JSONB	User-submitted data (cpf, cnh_number, etc.)
ln_invoice_id	TEXT	Lightning invoice ID from WDK Spark
ln_invoice	TEXT	BOLT11 invoice string for QR
amount_sats	INTEGER	Payment amount in satoshis

Column	Type	Description
`payment_status`	TEXT	pending | paid
`created_at`	TIMESTAMP	Creation timestamp

API Schema

Base URL: /api/v1

Auth

Method	Endpoint	Description
POST	`/auth/login`	Get access token
POST	`/auth/refresh`	Refresh token

POST /auth/login

Request: {email, password}
Response 200: {access_token, refresh_token, expires_in}

Tickets

Method	Endpoint	Description
POST	`/tickets`	Create ticket + LN invoice
GET	`/tickets`	List user's tickets
GET	`/tickets/:id`	Get ticket details
POST	`/tickets/:id/confirm-payment`	Confirm payment status

POST /tickets

Request: {operation_id, form_data: {cpf, cnh_number, cnh_mirror}}
Response 201: {ticket_id, ln_invoice, amount_sats}
Response 400: {error: "missing_required_fields", missing: [...]}

GET /tickets

Query: ?status=pending&limit=20&offset=0
Response 200: {tickets: [...], total: n}

GET /tickets/:id

Response 200: {id, operation_id, operation_name, form_data, ln_invoice, amount_sats, payment_status, created_at}

POST /tickets/:id/confirm-payment

Request: (empty body)
Response 200: {status: "paid"} or {status: "pending"}

Operations

Method	Endpoint	Description
GET	`/operations`	List all available operations- for now only driver license renewal
GET	`/operations/:id`	Get operation details

Error Codes

Code	HTTP	Description
`unauthorized`	401	Missing/invalid token

Code	HTTP	Description
`forbidden`	403	Not your ticket
`ticket_not_found`	404	Ticket doesn't exist
`missing_required_fields`	400	form_data validation failed
`invoice_creation_failed`	500	WDK Spark error

MCP Schema

Wrapper around the API, read-only tools exposed via MCP. Agent uses these for queries.
See: https://github.com/tadata-org/fastapi_mcp

Tools

Tool	Description
`list_operations`	List all available DETRAN operations
`get_operation`	Get operation details by ID or name
`list_tickets`	List tickets for a user (with filters)
`get_ticket`	Get ticket by ID

list_operations

Parameters: (none)
Returns: [{id, name, description, price, required_fields}]

get_operation

Parameters: operation_id (int) OR operation_name (string)
Returns: {id, name, description, price, required_fields}

list_tickets

Parameters: user_id (required), status (optional), limit (optional), offset (optional)
Returns: [{id, operation_name, payment_status, amount_sats, created_at}]

get_ticket

Parameters: ticket_id (uuid)
Returns: {id, operation_id, operation_name, form_data, ln_invoice, amount_sats, payment_status, created_at}

Toolbox Config (tools.yaml)

```
sources:
  detran-db:
    kind: postgres
    host: ${DB_HOST}
    port: 5432
    database: detran
    user: ${DB_USER}
    password: ${DB_PASSWORD}

tools:
  list_operations:
    source: detran-db
    kind: postgres-sql
    statement: SELECT id, name, description, price, required_fields FROM operations

  get_ticket:
    source: detran-db
    kind: postgres-sql
    statement: |
      SELECT t.*, o.name as operation_name
      FROM tickets t JOIN operations o ON t.operation_id = o.id
      WHERE t.id = $1
    parameters: [ticket_id]

toolsets:
  detran-read:
    - list_operations
    - get_operation
    - list_tickets
    - get_ticket
```

Example Agent Queries

- *"What services are available?"* → list_operations
- *"Show my pending tickets"* → list_tickets(user_id, status='pending')
- *"What's the status of ticket #abc?"* → get_ticket(ticket_id)
- *"How much does license renewal cost?"* → get_operation(name='driver_license_renewal')

Frontend Tools

Advertised to Agent via CopilotKit useFrontendTool.

Tool	Description
`start_driver_license_renewal`	Opens the CNH renewal form in UI