# GridPay — Functional and Technical Specifications

Version: 0.1  
Author: AYUMBA LUSA JONAS  
Date: 2025-09-01

## 1. Context and Objective

GridPay is an application designed to allow users to purchase electricity on demand.  
The user defines the amount of energy they want to consume (in kWh)...

## 2. Scope

**Included:**

* Automatic creation and calculation of an invoice / energy voucher.
* Online payment (one-time) and payment confirmation.
* IoT control to reconnect / cut off power based on payment and consumption.
* Transmission and storage of consumption measurements.
* User interface (invoice creation, status, history).
* Backend API and IoT broker (MQTT/Webhooks).

**Excluded (out of MVP):**

* Advanced dynamic pricing depending on grid load.
* Multi-rate management (time-based tariffs) and complex contracts.
* P2P energy marketplace/distribution.

## 3. Actors

* End-user
* Payment system (PSP)
* GridPay backend
* IoT broker / Device
* Administrator

## 4. Main Processes and Use Cases

1. **Invoice creation (pre-purchase)**
2. **Payment**
3. **Delivery and monitoring**
4. **Replenishment**

## 5. Detailed Scenarios

Examples: normal scenario, failed payment, offline device...

## 6. Functional Requirements (FR)

* **FR1 — Invoice creation**
* **FR2 — Validation/confirmation**
* **FR3 — Payment**
* **FR4 — IoT command**
* **FR5 — Consumption measurements**
* **FR6 — Automatic cut-off**
* **FR7 — History**
* **FR8 — Reuse (replenishment)**
* **FR9 — Error management**
* **FR10 — Notifications**

## 7. Non-Functional Requirements (NFR)

* **NFR1 — Payment security**
* **NFR2 — Authentication/Authorization**
* **NFR3 — Availability**
* **NFR4 — Real-time responsiveness**
* **NFR5 — Scalability**
* **NFR6 — Traceability**
* **NFR7 — Data confidentiality**

## 8. Proposed Technical Architecture

* **Frontend:** Web/Mobile
* **Backend:** REST/GraphQL API
* **Database:** PostgreSQL + Time-series DB
* **IoT Broker:** MQTT/WS
* **Device:** MQTT/HTTPS with TLS
* **PSP:** Stripe/PayPal/local PSP
* **Deployment:** Docker/Kubernetes

## 9. Protocols and IoT Integration

MQTT over TLS, structured topics, authentication via certificates or tokens.

## 10. Data Model (extract)

User, Device, Invoice, Payment, ConsumptionReading.

## 11. API (examples)

* POST /invoices
* GET /invoices/{id}
* POST /payments/{invoice\_id}/start
* POST /webhooks/psp
* POST /devices/{deviceId}/telemetry
* POST /devices/{deviceId}/command

## 12. Error and Edge Case Management

Confirmed payment but offline device, over-consumption, partial payment, duplicate PSP notifications, network recovery.

## 13. Testing and Validation

Unit, integration, end-to-end, load, security, functional acceptance.

## 14. Deployment & Monitoring

CI/CD, monitoring (Prometheus/Grafana), alerting.

## 15. MVP (Minimum Viable Product)

Minimal frontend, invoices + PSP backend, IoT simulator, basic consumption storage.

## 16. Roadmap and Evolutions

Dynamic pricing, recurring payments, advanced admin interface, analytics, multi-device support.

## 17. Acceptance Criteria

Energy purchase → payment → IoT delivery → cut-off when quota is reached → traceable payment.

## 18. Annexes

User messages, default values (telemetry interval, delivery timeout, kWh price).