



Dar El Behy

Smart Home Automation

Louey Dridi Dhia eddin chedly Mohamed Belgacem Achref Msekni
Mohamed Amine Saoudi

Advisor: Dr. Ameni Azouz

Tunis Business School

18/12/2025

Table of Contents

1. Introduction
2. Technologies and Tools
3. System Architecture
4. Core Components
5. User Interface
6. Features
7. UML Diagrams
8. Security
9. Testing
10. Installation and Usage
11. Scalability
12. Conclusion



Introduction

- **Dar El Behy** is a JavaFX-based smart home control system.
- Centralizes control of multiple smart devices through a graphical interface.
- Supports voice commands, real-time energy monitoring, and prayer times integration.
- Designed for usability, security, and future extensibility.



Technologies and Tools

- **Language:** Java 25 (Object-Oriented Programming).
- **UI Framework:** JavaFX 25.0.1 with WebView and CSS.
- **APIs:** Aladhan (Prayer Times), IP-API (Geolocation), Web Speech API.
- **Storage:** JSON for configuration, CSV for energy history.

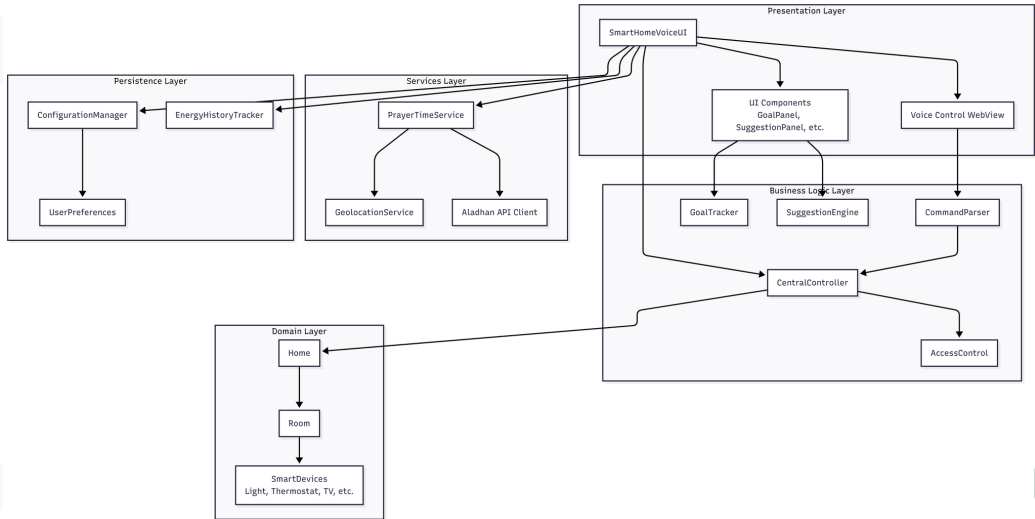


System Architecture

- Layered architecture with clear separation of concerns.
- Presentation (UI components), Business Logic (Controllers), Domain (Core entities & analytics), Services (External API integrations), and Persistence layers (Data storage and retrieval).
- CentralController acts as a facade for system operations.
- Architecture supports scalability and maintainability.



System Architecture



Core Components

- **SmartDevice** abstract base class with 11 concrete device types.
- **CentralController** orchestrates commands and device interactions.
- **CommandParser** processes natural language voice commands.
- **GoalTracker** monitors energy consumption and goals.



User Interface

- JavaFX-based interface with room-oriented device control.
- Integrated voice command panel and activity console.
- Energy dashboard with goals and real-time monitoring.
- Parent and Child modes with PIN-based access control.



Interface

Dar El Behy Smart Home Control System

Dar El Behy Smart Home Access Mode: **Parent Mode**

Voice Control

Dar El Behy Smart Home
Intelligent home automation with voice control

Voice Command Center

Quick Actions:

- List All Devices
- Show Energy Usage
- Low Power Mode
- Save Configuration
- Export Report
- Open in Browser
- + Add Room

Device Controls

LIVING ROOM + Add Device

Living Light 1 OFF

Living Light 1 (Light) [OFF] brightness=80 color=#FFFFFF

Turn On Turn Off

Brightness: 0 25 50 75 100

Living Light 2 OFF

Living Light 2 (Light) [OFF] brightness=80 color=#FFFFFF

Turn On Turn Off

Brightness: 0 25 50 75 100

Energy Dashboard

Total: 0,01 kW

Est. Cost: \$0,00/hr (\$0,02/day)

Device Breakdown:

- living room - Living Room TV
5 W (100,0%)
- kitchen - Main Oven
0 W (0,0%)
- kitchen - Kitchen Counter Light
0 W (0,0%)

Energy Goals + New Goal

Overall Progress

0,0 / 350,0 kWh (0,0%)

Activity Log

- Main Garage Door (GarageDoor) [CLOSED]

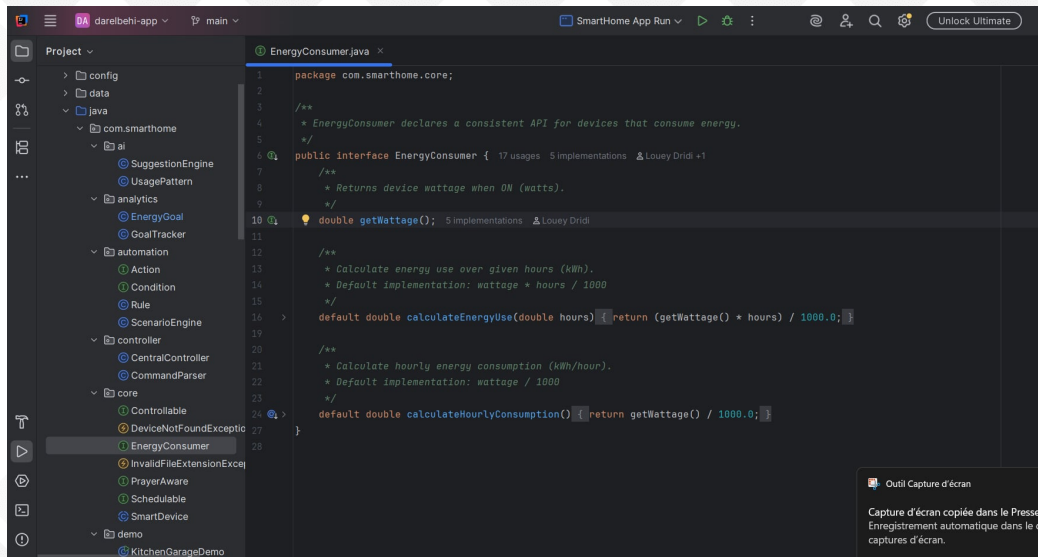
Tip: Use voice commands or device controls to manage your home!
✓ Voice recognition ready

Key Features

- **Voice Control:** Web Speech API processes voice commands through a WebView and executes them via the CentralController.
- **Energy Management:** Real-time energy tracking every 5 minutes with goals, alerts, and CSV reports.
- **Prayer Times & AI:** Automatic prayer times via Aladhan API and AI-based energy saving suggestions.



Code snippets



The screenshot shows an IDE window with a project named "darebehi-app" and a main branch. The project structure on the left includes folders like "config", "data", "java", "com.smarthome", "ai", "analytics", "automation", "controller", "core", and "demo". The "EnergyConsumer.java" file is open in the editor, showing the following code:

```
1 package com.smarthome.core;
2
3 /**
4  * EnergyConsumer declares a consistent API for devices that consume energy.
5  */
6 public interface EnergyConsumer { 17 usages 5 implementations Louey Dridi +1
7     /**
8      * Returns device wattage when ON (watts).
9      */
10     double getWattage(); 5 implementations Louey Dridi
11
12     /**
13      * Calculate energy use over given hours (kWh).
14      * Default implementation: wattage * hours / 1000
15      */
16     default double calculateEnergyUse(double hours) { return (getWattage() * hours) / 1000.0; }
17
18     /**
19      * Calculate hourly energy consumption (kWh/hour).
20      * Default implementation: wattage / 1000
21      */
22     default double calculateHourlyConsumption() { return getWattage() / 1000.0; }
23 }
24
25 }
```

At the bottom right, there is a notification for "Outil Capture d'écran" (Screenshot tool) with the text: "Capture d'écran copiée dans le Presse-papier. Enregistrement automatique dans le dossier captures d'écran."

Code snippets

```

package com.smarthome.devices;

import ...

public class Fridge extends SmartDevice implements EnergyConsumer {
    private double internalTemperature = 4.0;
    private final List<String> groceryList = new ArrayList<>();

    public Fridge(String id, String name) {
        super(id, name);
        this.on = true; // Fridge is generally always on
    }

    @Override public void turnOn() { /* Always on */ }
    @Override public void turnOff() { /* Always on */ }

    public void setInternalTemperature(double temp) { this.internalTemperature = Math.max(1.0, Math.min(6.0, temp)); }

    public double getInternalTemperature() { return internalTemperature; }

    public void addToGroceryList(String item) {
        String normalizedItem = item.trim();
        if (!normalizedItem.isEmpty() && !groceryList.contains(normalizedItem.toLowerCase())) {
            groceryList.add(normalizedItem.toLowerCase());
        }
    }

    public String getGroceryList() {
        if (groceryList.isEmpty()) return "Grocery list is empty.";
        return String.join(", ", groceryList);
    }
}

```

UML Overview

Use Case Diagram

Defines interactions between Parent, Child, System, and External APIs.

Sequence Diagrams

Illustrates voice command processing, energy tracking, and prayer time loading.

Class Diagram

Shows SmartDevice hierarchy, interfaces, and Home-Room composition.



Security Considerations

Parent and Child modes with PIN-protected access to critical devices.

Rate Limiting

Prevents excessive API calls and brute-force PIN attempts.

Future Improvements

PIN hashing, encrypted configuration files, and multi-user authentication.



Testing Strategy

Automated Testing

JUnit tests for CentralController, CommandParser, and Home/Room logic.

Coverage Areas

Device state management, energy calculations, and command parsing.

Manual Testing

Voice recognition, UI responsiveness, and prayer times accuracy.



Installation and Usage

- JDK 25, JavaFX SDK 25.0.1, and an internet connection.
- Application launched using `run.bat` after JavaFX configuration.

Voice Commands

“Turn on all lights in living room”, “Set thermostat to 22”.



Scalability

- **SOLID Principles**

Modular and extensible system design

- **Data Persistence**

Decoupled and scalable data access



Conclusion

A complete JavaFX smart home system with voice control, energy analytics, and prayer times.

11 device types, layered architecture, persistent configuration.

Future Work

Cloud sync, mobile app, IoT integration, and machine learning.

