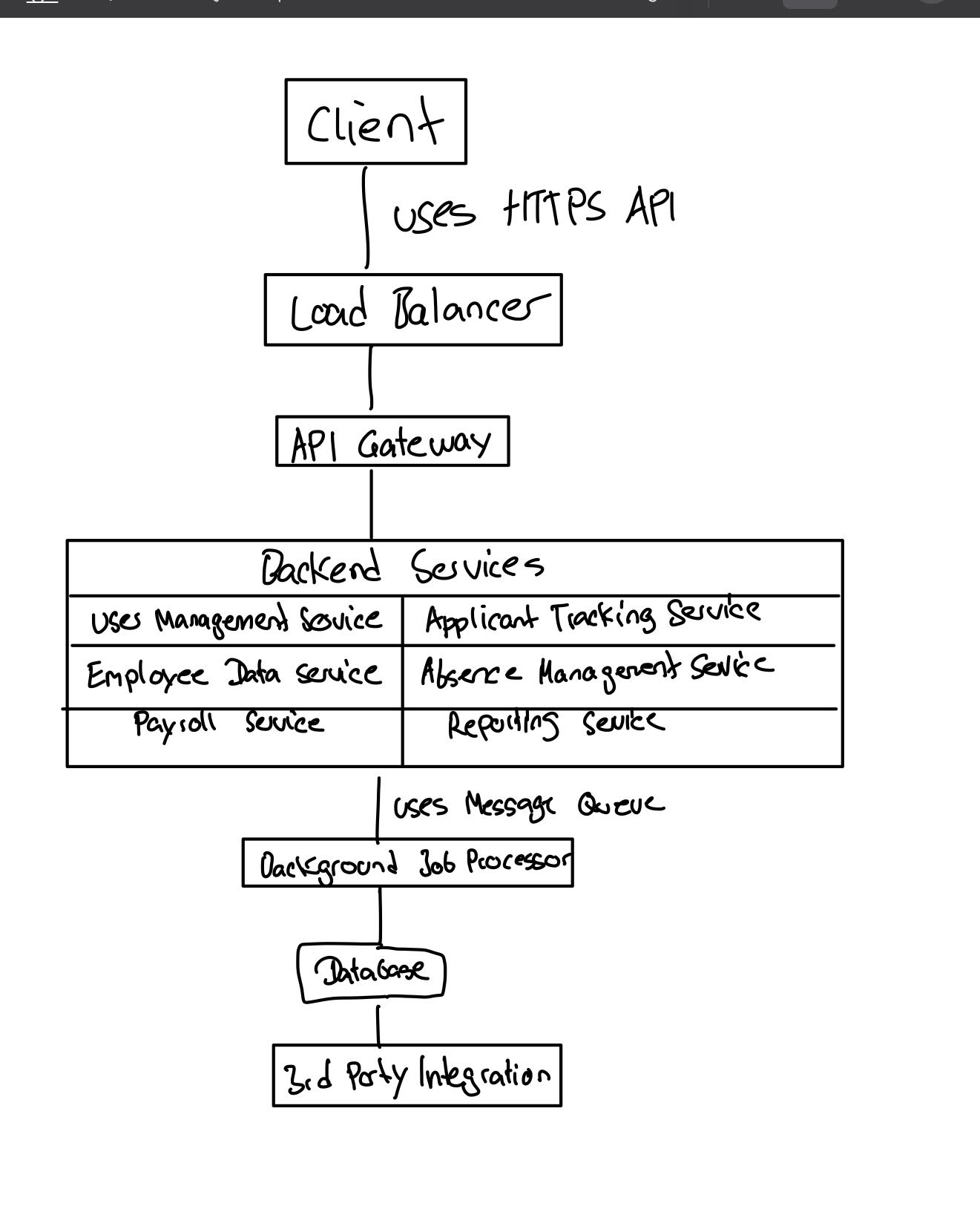
**Exercise 2**

**1. personio.com Analysis**

* **Functionality:** Personio.com provides software (SaaS - Software as a Service) that helps companies manage their employees. The main things it does are:
  + **Applicant Tracking:** Helps to find and hire new employees.
  + **Employee Data Management:** Keeps records of employee information.
  + **Absence Management:** Tracks when employees are on vacation or sick.
  + **Payroll Preparation:** Helps to calculate employee paychecks.
  + **Reporting and Analytics:** Provides information about employees.
* **Structural Architecture:**  
  
* **Description:**
  + **Client:** Users use the software through a website or a mobile app.
  + **Load Balancer:** Makes sure the system can handle many users at once.
  + **API Gateway:** A single entry point for requests, directing them to the right service.
  + **Backend Services:** Small, independent programs that handle specific tasks.
  + **Message Queue:** Helps services communicate reliably, especially for background tasks.
  + **Background Job Processor:** Runs tasks that take a long time without slowing down the user.
  + **Database:** Stores all the data for the application.
  + **3rd Party Integrations:** Allows Personio to work with other software.
* **Architectural Style:** This is a **Microservices Architecture**.
  + Services are small and can be updated independently.
  + Services communicate using APIs.
  + Each service manages its own data.
  + If one service fails, the whole system might still work.

**2. Yahyavi PDF Analysis (Pages 1-20)**

* **Architectural Style Description:** The Yahyavi PDF mainly discusses the **Peer-to-Peer (P2P) architectural style**, especially for online games with many players.
  + It explains that P2P systems share the work (like calculations and network traffic) among the players' computers (peers) instead of relying on one central computer (server).
  + P2P can handle many players and can be cheaper than using big servers.
  + However, P2P also has problems, like making sure everyone agrees on what's happening in the game, preventing cheating, and dealing with players leaving the game.
  + The paper also talks about "Interest Management," which is a technique used in P2P games to manage what information each player needs to know. For example, a player only needs to know what's happening near their character in the game. Zoning is a common technique for this.
  + **Information Location:** This description is based on sections throughout pages 1-20 of the Yahyavi PDF. Key concepts are introduced in the "Introduction" (page 1) and further explained in sections discussing scalability, cost-effectiveness, challenges, and interest management.
* **Comparison to Other Styles:**
  + **Client-Server:** The Yahyavi PDF compares P2P to the traditional client-server model (discussed on pages 2-3), where one central server runs the game. Client-server can have problems with handling many players and can be expensive.
  + **Multiserver:** The paper also mentions multiserver architectures (also on pages 2-3), where the game is divided among multiple servers. P2P ideas can sometimes be used to improve or work with multiserver systems.
  + **Information Location:** These comparisons are found primarily in the "Introduction" and the sections that describe the limitations of traditional architectures and the benefits and challenges of P2P.

**3. Ticket Machine Architectural Views**

* **Logical View:**
  + Components: User Interface, Ticket Logic, Fare Calculation, Payment, Ticket Printer, Database Connection, Communication.
  + Relationships: The User Interface uses the Ticket Logic. The Ticket Logic uses Fare Calculation, Payment, and Database Connection. The Communication connects to a central system.
* **Process View:**
  + Processes: User Interaction, Ticket Purchase, Payment Authorization, Ticket Printing, System Update.
  + Concurrency: The machine can handle multiple users at the same time.
  + Communication: Processes communicate inside the machine and with a central system.
* **Development View:**
  + Modules: UI Module, Business Logic Module, Data Access Module, Communication Module.
  + Subsystems: The modules can be divided further, like UI (screen, buttons), Business Logic (rules for tickets), etc.
  + Dependencies: How the modules depend on each other.
* **Physical View:**
  + Hardware: Touchscreen, Printer, Card Reader, Computer, Memory, Network Connection.
  + Network: Connection to a central computer system.
  + Deployment: How the software is put onto the hardware.

**4. Architectural Patterns**

* **a) Whistleblowing System:**
  + **Recommended Pattern:** **Layered Architecture**.
  + **Why:**
    - Separation: Different layers handle different things (user interface, rules, data), making it easier to manage.
    - Security: Layers can add security (like who can see what) at different levels.
    - Easy to Change: Changing one layer is less likely to break other parts of the system.
* **b) Video Conferencing System:**
  + **Recommended Pattern:** **Client-Server Architecture** (with some **Peer-to-Peer**).
  + **Why:**
    - Control: A server can manage who joins the call, security, etc.
    - Scalability: Servers can be added to handle more users.
    - Efficiency: For sending video and audio, direct connections between users (P2P) can be faster.
* **c) GPS Tracker for Cats:**
  + **Recommended Pattern:** **Event-Driven Architecture**.
  + **Why:**
    - Real-time: The tracker sends out "events" (location updates) that need to be processed quickly.
    - Scalability: Can handle many cats sending updates.
    - Flexibility: It's easy to add new features that react to the location events (like alerts if the cat leaves a certain area).