

# Case Study #3

Memi Lavi  
[www.memilavi.com](http://www.memilavi.com)





GROCECOLL

# GROCECOLL

---

- Grocery collection service
- Allows customers to create shopping lists that get collected and delivered by GroceColl's employees
- Available world-wide



# GROCECOLL

---

- Employees have dedicated tablets displaying the list
- We need to design the collection side of the system
  - The customer side is already developed





# Requirements

## Functional

What the system should do

1. Web Based
2. Tablets receive list to be collected
3. Employees can mark items as collected or unavailable
4. When collection is done, the list should be transferred to payment engine
5. Offline support is a must

## Non-Functional

What the system should deal with



## NFR - What We Ask

1. *"How many expected concurrent users?"* 200
2. *"How many lists will be processed per day?"* 10,000
3. *"What is the average size of a shopping list?"* 500KB



## NFR - What We Ask

4. *"Do we need offline support?"*

Yes!

5. *"What is the desired SLA?"*

Highest Possible

6. *"How do lists arrive to the system?"*

Queue



## Data Volume

- 1 List = 500KB
  - 10,000 lists / day = 5GB / day
- => ~2TB / year





# Requirements

```
graph TD; R[Requirements] --> F[Functional]; R --> NF[Non-Functional];
```

## Functional

What the system should do

1. Web Based
2. Tablets receive list to be collected
3. Employees can mark items as collected or unavailable
4. When collection is done, the list should be transferred to payment engine
5. Offline support is a must

## Non-Functional

What the system should deal with

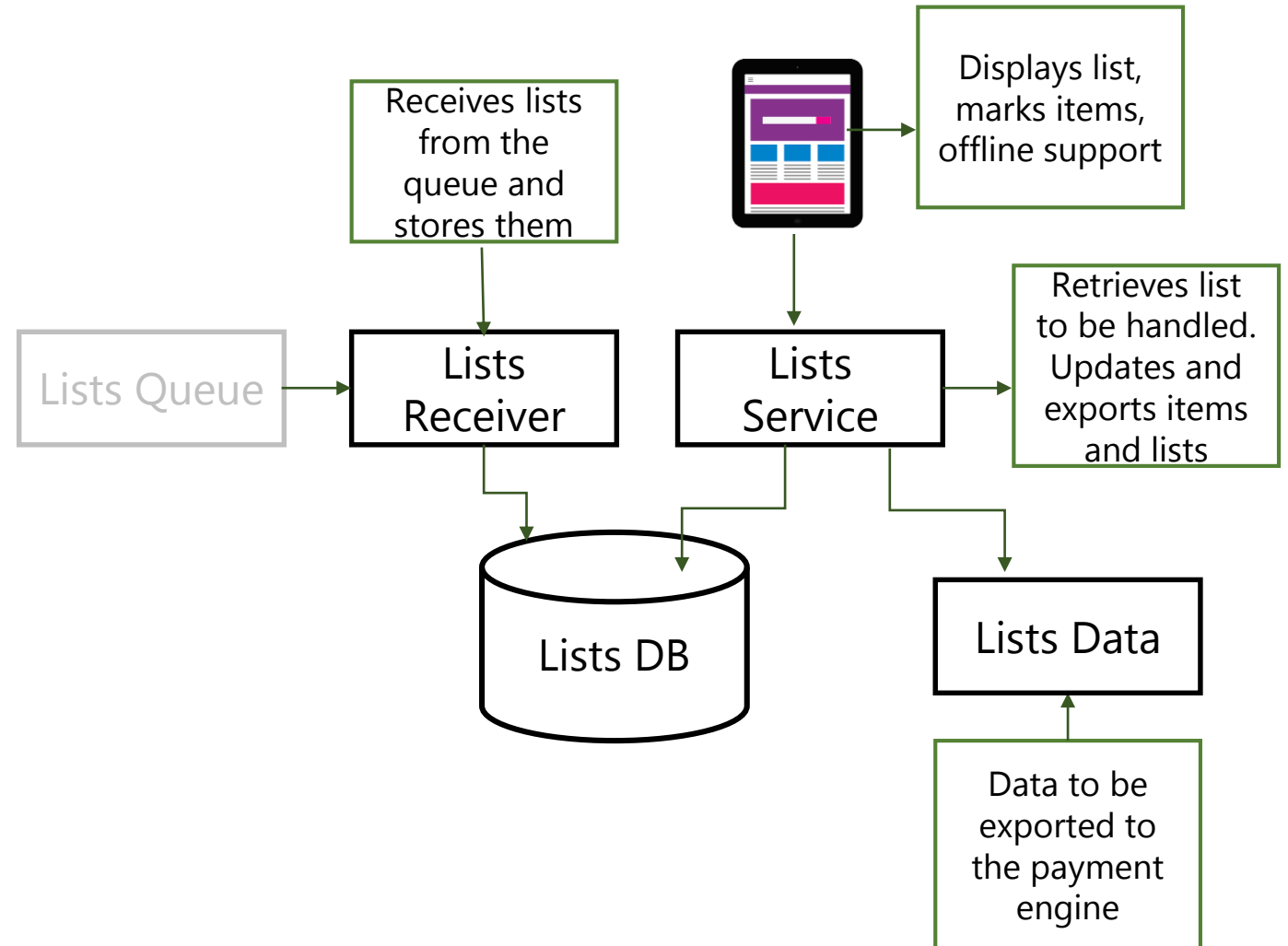
1. 200 Concurrent users
2. 10,000 lists/day
3. Yearly volume: 2TB
4. High SLA
5. Offline support



# Components

Based on requirements:

1. Employees have tablets
2. Offline support
3. Retrieve lists
4. Mark Items
5. Export list to payment engine

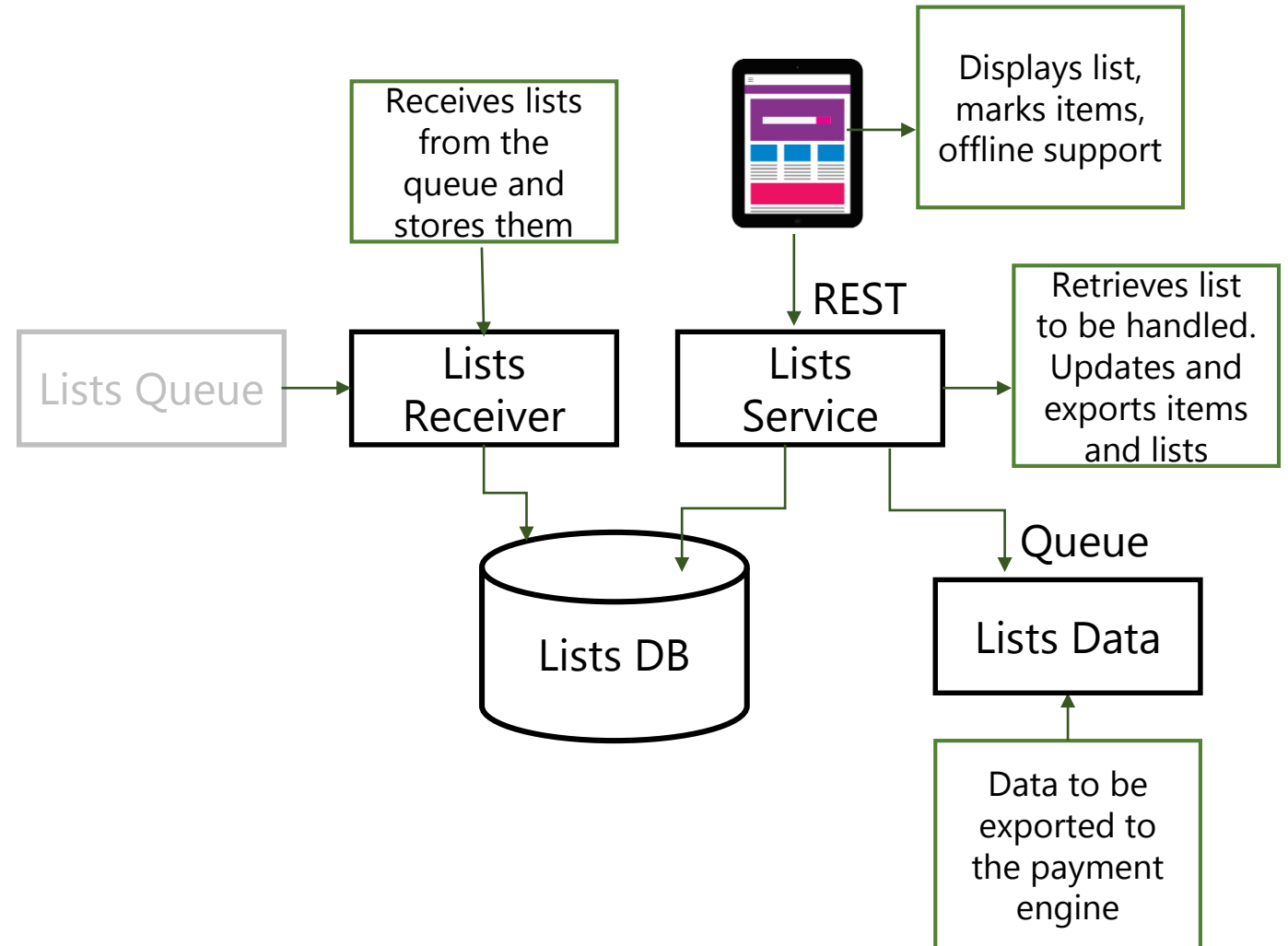




# Messaging

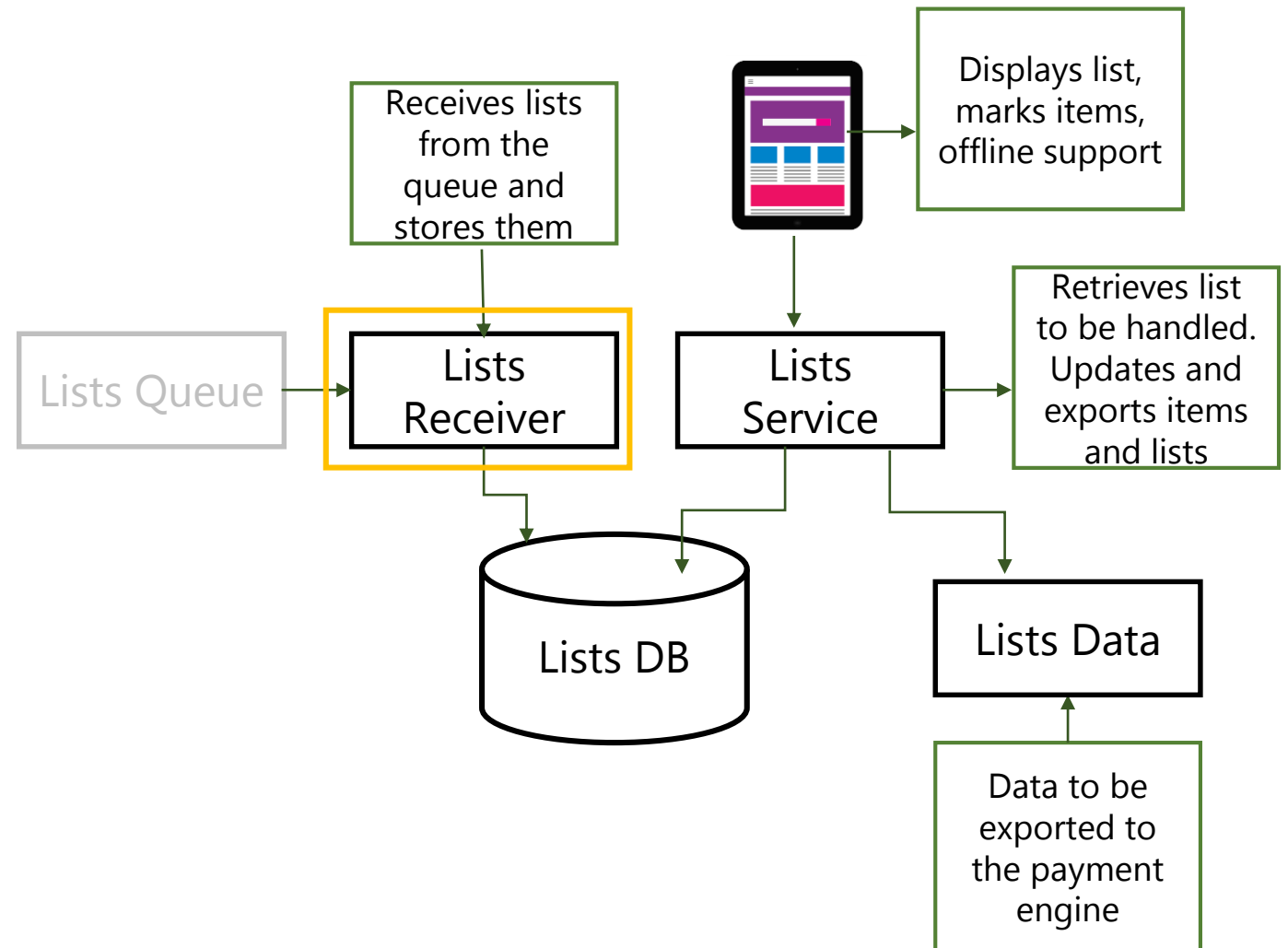
Based on requirements:

1. Employees have tablets
2. Offline support
3. Retrieve lists
4. Mark Items
5. Export list to payment engine





# Components





## Lists Receiver

What it does:

- Receives shopping lists to be handled from queue
- Stores the lists in the datastore



## Application Type

- Web App & Web API ❌
- Mobile App ❌
- Console ✔️
- Service ✔️
- Desktop App ❌



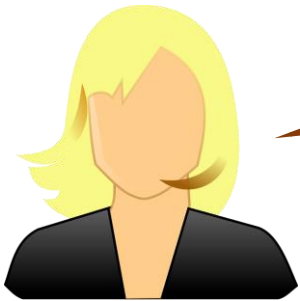
## Technology Stack

### Considerations:

- Should be able to connect to queue
- Not much else...



# Technology Stack



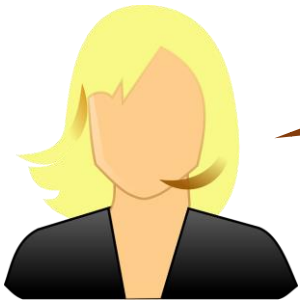
We're basically a Java shop, and our database of choice is MySQL.

Java is a perfect fit for this task, so we'll go with it.





# Technology Stack



We're basically a Java shop, and our database of choice is MySQL.

What about database?

- Our data is relational, and MySQL is a relational DB
- Expected volume is 2TB/Year which is a lot
  - But can utilize partitioning
  - So...



## Technology Stack





# Architecture

Queue Receiver

Business Logic

Data Access

Data Store



# Lists Receiver Redundancy

Consumer Group

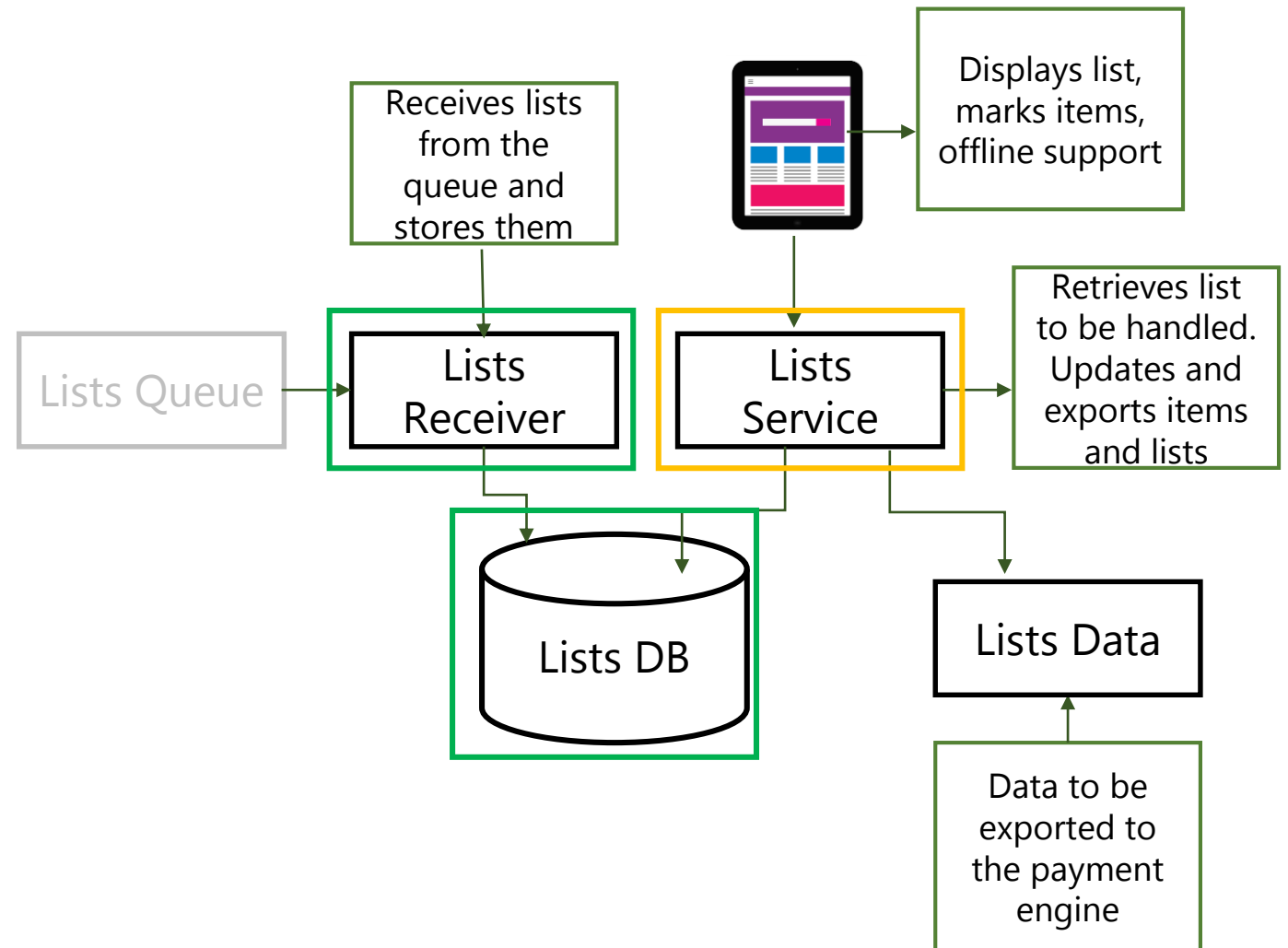
Lists Receiver

Lists Receiver

Lists Receiver



# Components





## Lists Service

What it does:

- Allows employees to query lists
- Marks items in list
- Exports payment data



## Application Type

- Web App & Web API ✓
- Mobile App ✗
- Console ✗
- Service ✗
- Desktop App ✗



# Technology Stack







# Architecture

Service Interface

Business Logic

Data Access

Data Store



## API

- Get next list to be processed (by location)
- Mark item as collected / unavailable
- Export list's payment data

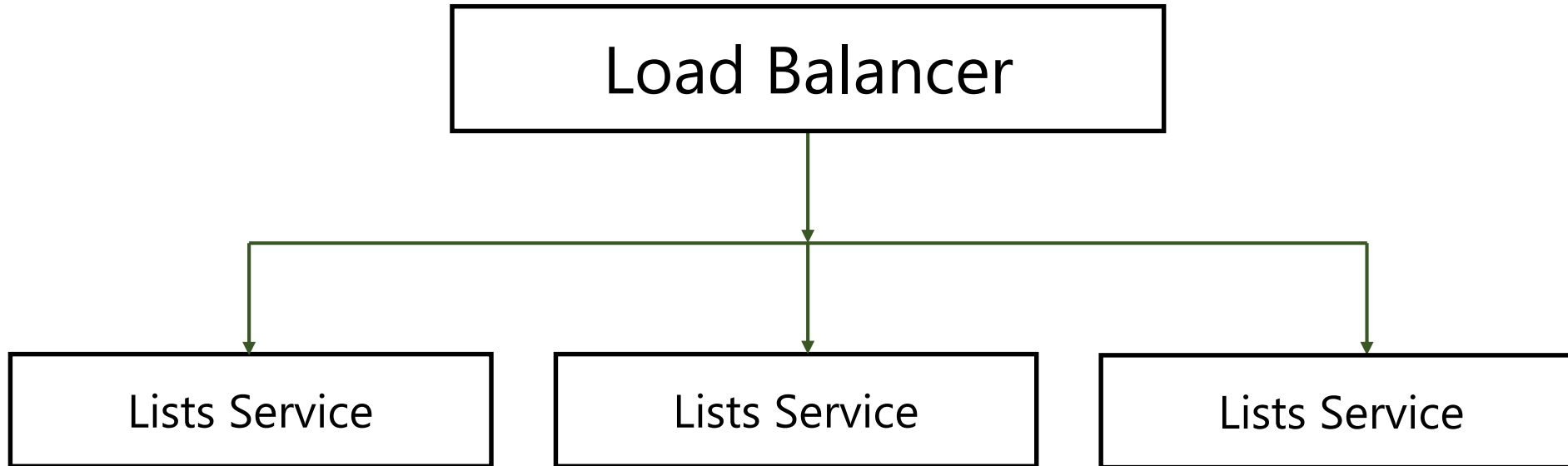


# API

Functionality	Path	Return Codes
Get next list to be processed	GET /api/v1/lists/next?location=...	200 OK 400 Bad Request
Mark item as collected / unavailable	PUT /api/v1/list/{listId}/item/{itemId}	200 OK 404 Not Found
Export list's payment data	POST /api/v1/list/{listId}/export	200 Ok 404 Not Found

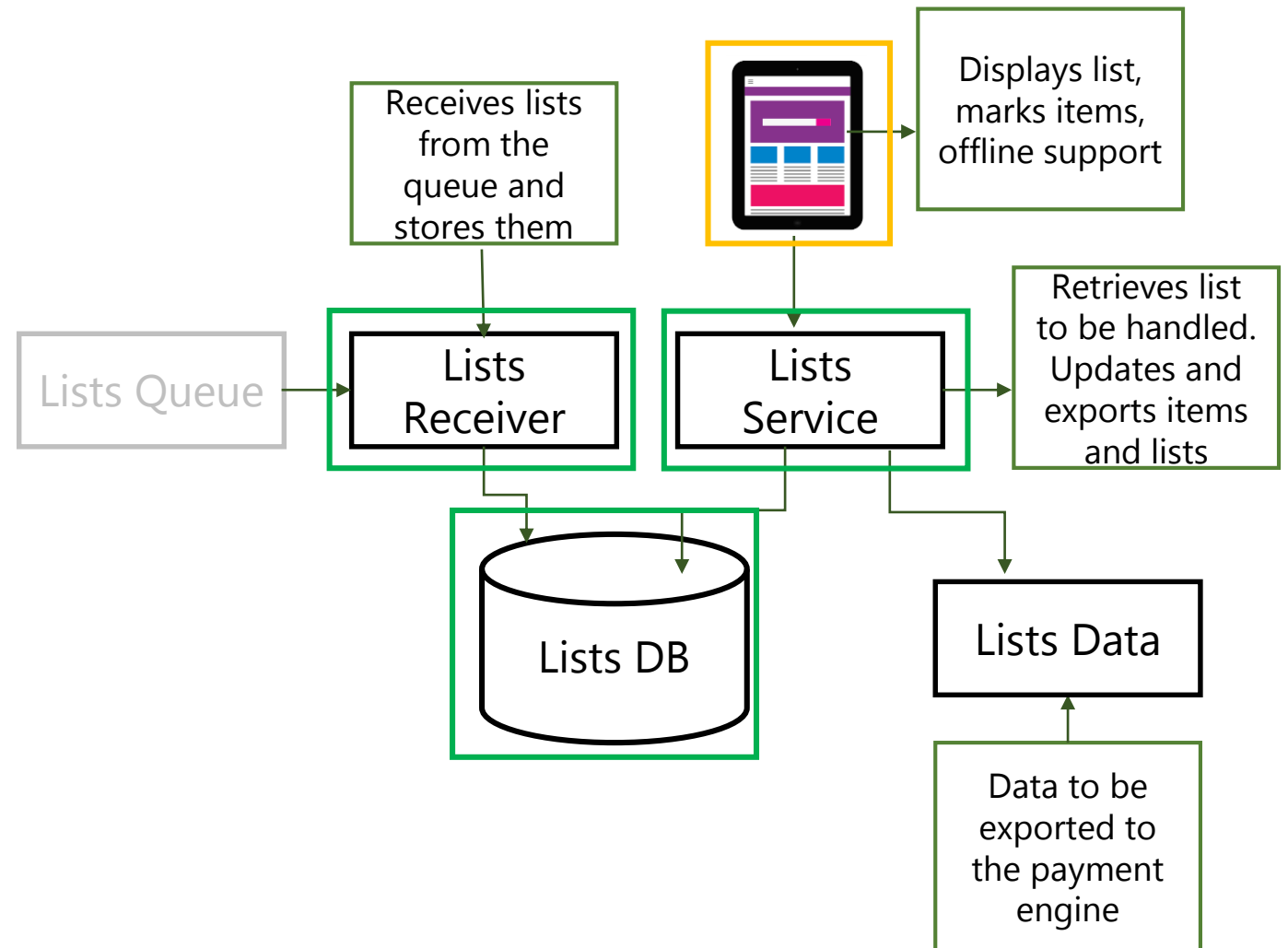


# Lists Service Redundancy





# Components





## Front End

What it does:

- Displays shopping list
- Marks items as unavailable / collected
- Sends list to payment system
- Supports offline mode



## Application Type

- Web App & Web API ✗
- Mobile App ✓
- Console ✗
- Service ✗
- Desktop App ✓



## Technology Stack

Need to decide between:

### Desktop, windows based (WPF)

- Supports all OS functionalities
- Utilizes other apps on the machine (ie. DB)
- Requires setup, Windows

### Web based (Electron, React Native)

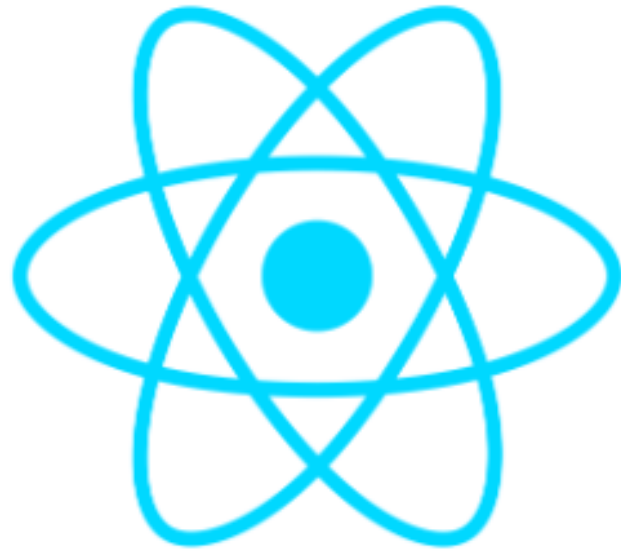
- Limited functionality
- Cannot use other apps
- Fully compatible with other form factors (phones, etc.)
- No setup required
- Cheaper hardware





## Technology Stack

Need to decide between:



React Native

### Web based (Electron, React Native)

- Limited functionality
- Cannot use other apps
- Fully compatible with other forms (phones, etc.)
- No setup required
- Cheaper hardware

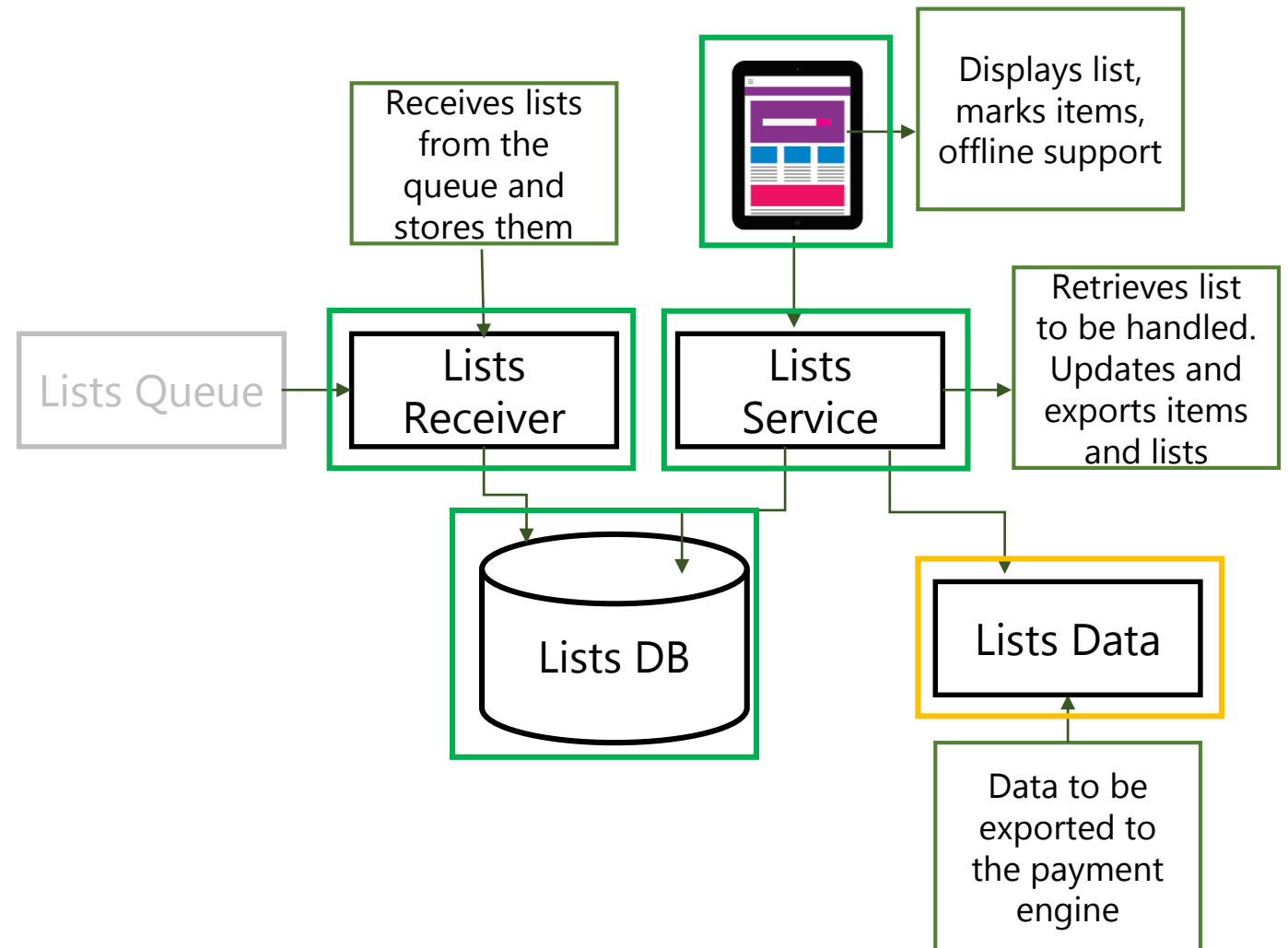


## Front End Redundancy

Not  
Relevant...



# Components





## Export Lists Data

What it does:

- Used to send shopping lists' data to payment system
- Basically – a queue



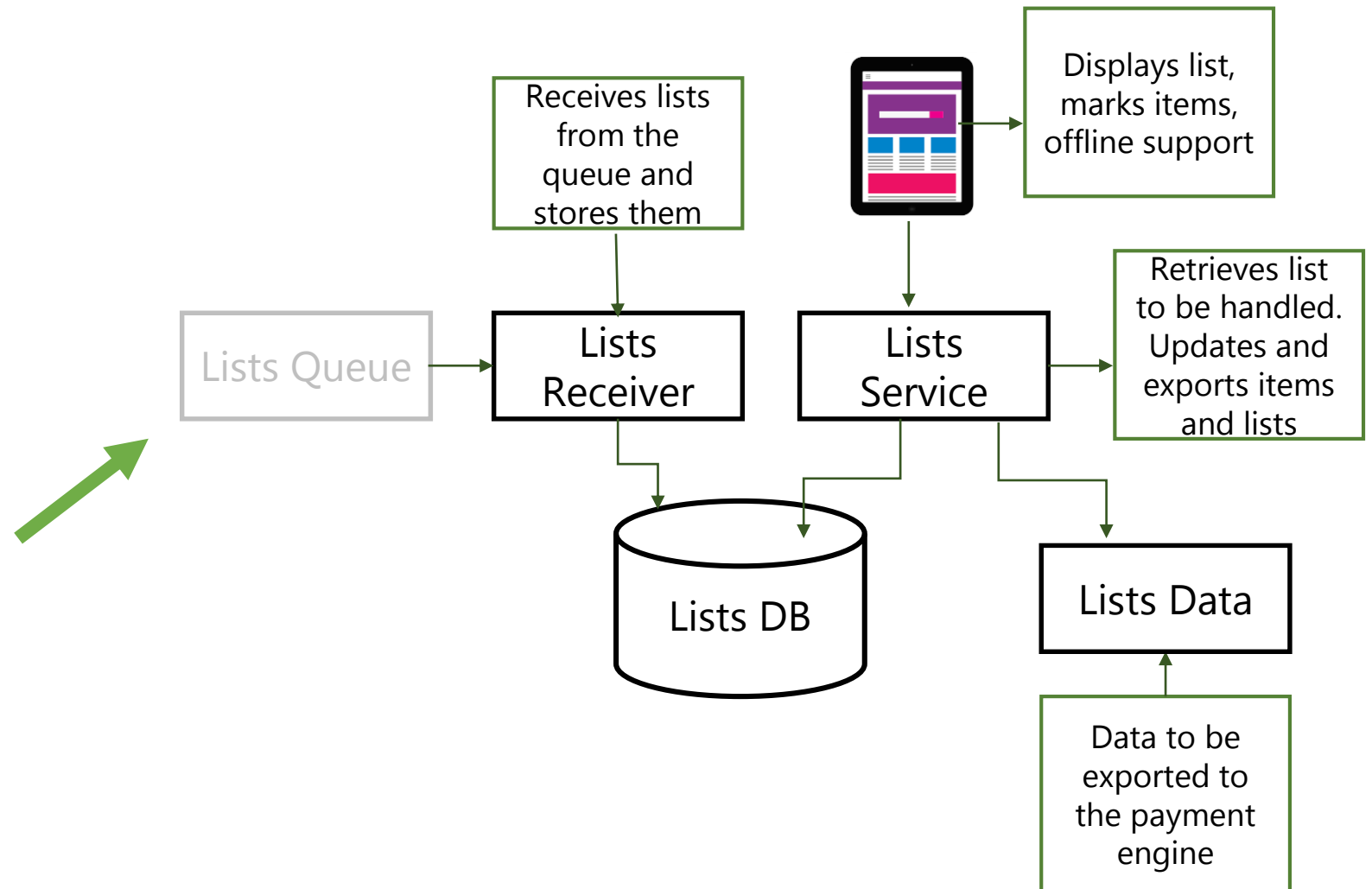
## Export Lists Data- Questions

1. Is there an existing queue mechanism in the company?
2. Develop our own or use 3<sup>rd</sup> party?

**Yes**

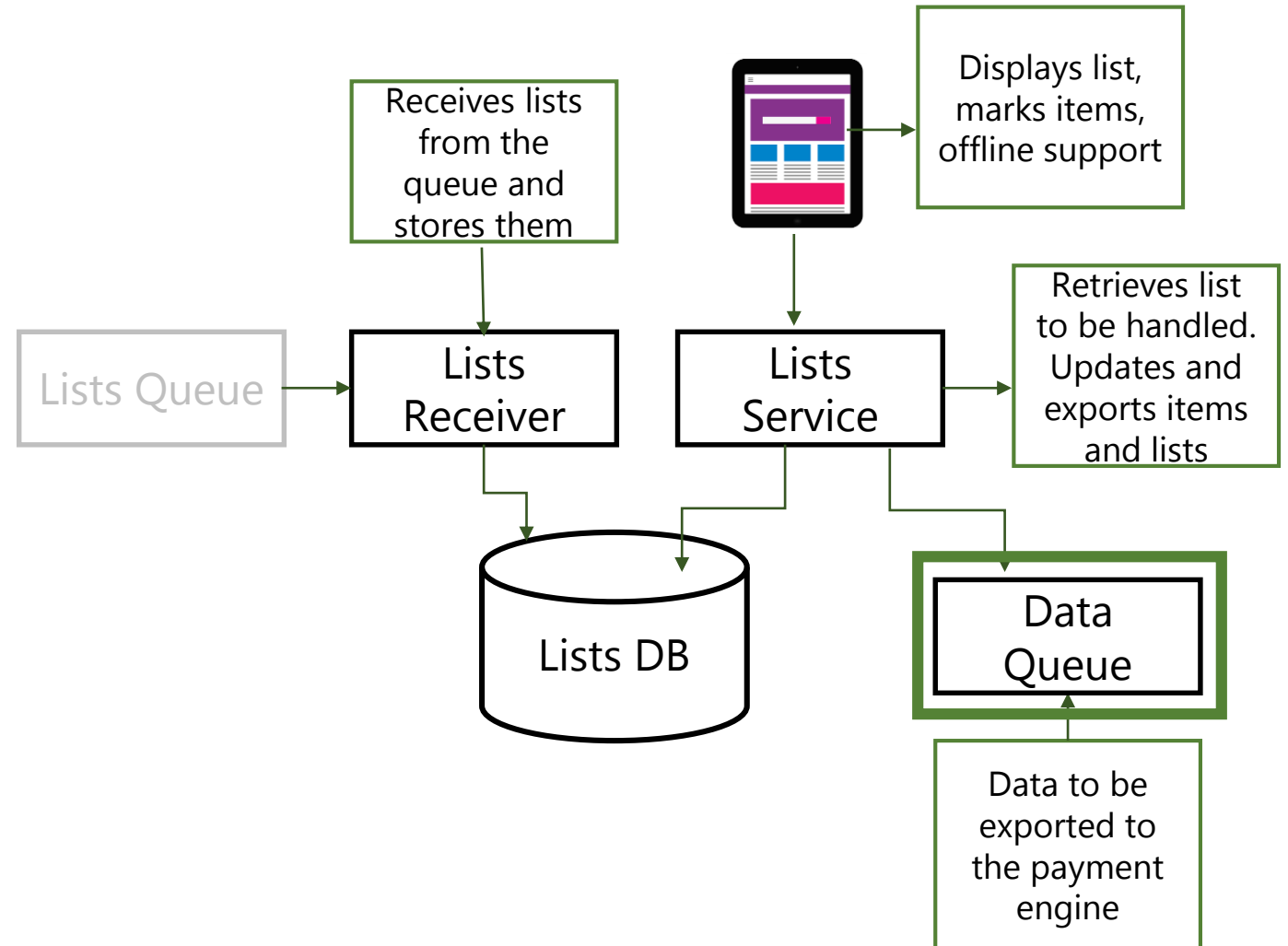


# Components



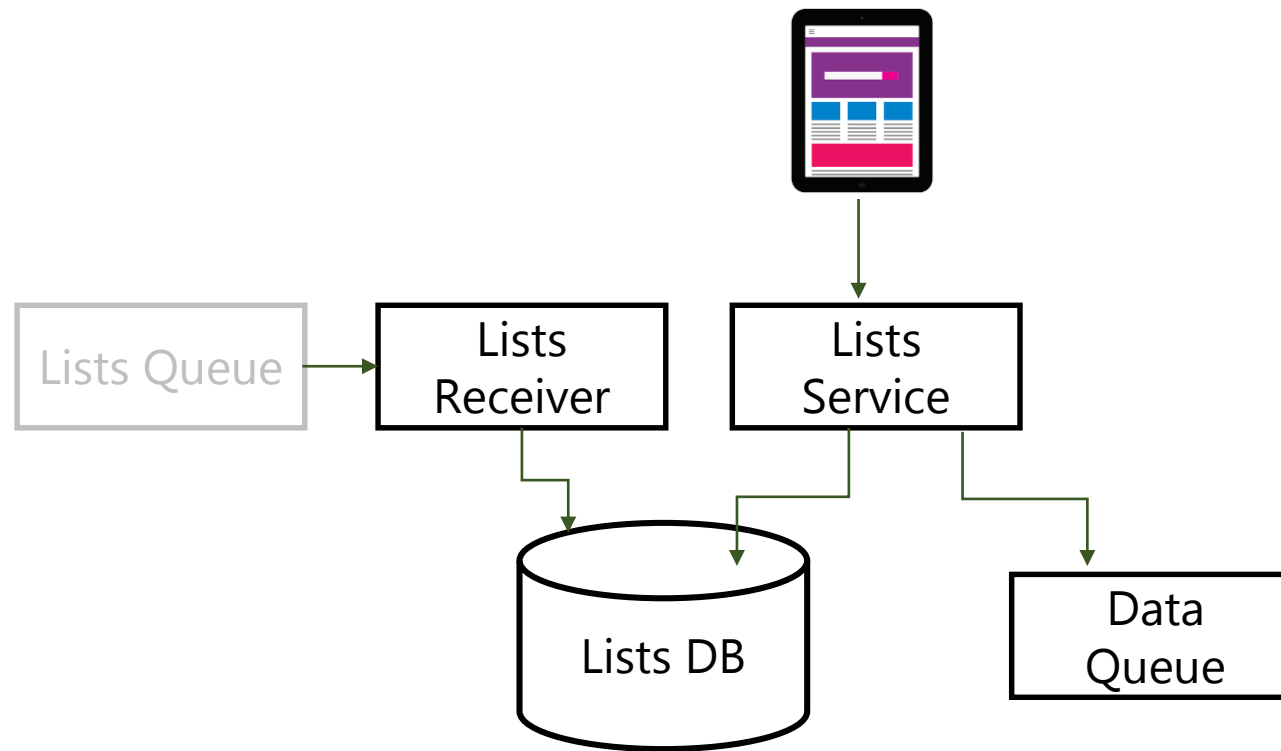


# Components





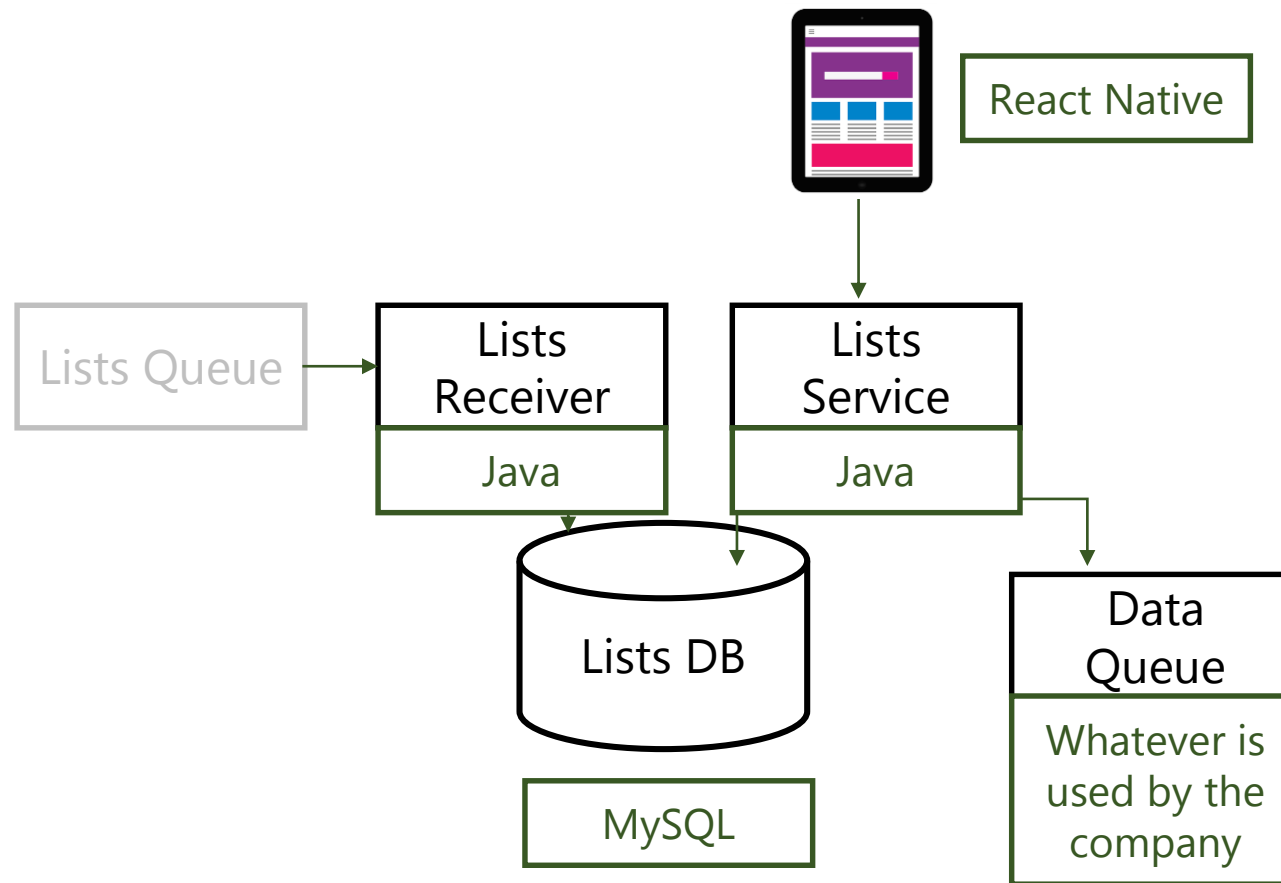
# Logic Diagram







# Technical Diagram





# Physical Diagram

