*Adrenaline Bikes ERP*

*Team: ; DROP TABLE TEAMS; --*

**SOFTWARE ARCHITECTURE DOCUMENT**

Version 1.0

24/01/21

**VERSION HISTORY**

| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| --- | --- | --- | --- | --- | --- |
| 1.0 | Mounceph Morssaoui Laetitia Segaux,  Dina Khalesi,  Nicolas Macbeth,  Yordan Pramatarov,  Alexander Shevchenko | 02/02/21 | Logan McMullen | 03/02/21 | Initial Draft of the SAD |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**TABLE OF CONTENTS**

[**INTRODUCTION**](#_heading=h.3znysh7) **1**

[1.1 Identifying Information](#_heading=h.2et92p0) 1

[1.2 Supplementary Information](#_heading=h.tyjcwt) 1

[1.2.1 Scope](#_heading=h.3dy6vkm) 1

[1.2.2 Authors](#_heading=h.1t3h5sf) 1

[1.3 Other Information](#_heading=h.2s8eyo1) 1

[1.3.1 System or Architecture Overview](#_heading=h.4d34og8) 1

[**STAKEHOLDERS AND CONCERNS**](#_heading=h.17dp8vu) **2**

[Stakeholders and concerns](#_heading=h.3rdcrjn) 2

[Concern-stakeholder traceability](#_heading=h.2lwamvv) 6

[**REQUIREMENTS**](#_heading=h.2zbgiuw) **8**

[Non-functional Requirements](#_heading=h.1egqt2p) 8

[Functional Requirements](#_heading=h.3ygebqi) 9

[**VIEWPOINTS AND VIEWS**](#_heading=h.2dlolyb) **10**

[Scenario view](#_heading=h.sqyw64) 11

[Logical View](#_heading=h.3cqmetx) 12

[Process View](#_heading=h.1rvwp1q) 13

[Development View](#_heading=h.4bvk7pj) 15

[Physical View](#_heading=h.2r0uhxc) 17

[**REFERENCES**](#_heading=h.1664s55) **19**

# **INTRODUCTION**

## 1.1 Identifying Information

The goal of this report is to present the software architecture choices we made regarding the development of a bike store ERP. The goal of the application is to provide software to bike store owners that allows them to manage their business processes with ease. The system was built using a react frontend, an Elixir Phoenix backend and a PostgreSQL database.

## 1.2 Supplementary Information

#### **1.2.1 Scope**

Due to the short duration of this class and the size of the team, the scope of this project is limited to the following constraints:

The system will only be available as a web-app.

The system will not be available as on premises software

No single tenant database architecture

#### **1.2.2 Authors**

This document was authored by team 10 from the SOEN-390 Winter 2021 semester.

## 1.3 Other Information

## 1.3.1 System or Architecture Overview

The frontend of our web-application is built in react as a static frontend that doesn’t make use of any templating language. React is used to receive information from the server via http calls. We made this choice because it decouples the two components from each other and allows us to extend to other frontends such as a mobile app more easily in the future.

On the backend, we have an Elixir Phoenix server. We decided to use Phoenix because it simplifies many aspects of web development by acting as both an http server that accepts HTTP POST connections and socket connections.

We use an MVC architecture, where we have for example a User model that can be modified by a controller when a request is received at a certain endpoint. The change is then propagated to our PostgreSQL database. We mainly interact with our database through the Phoenix ORM. On top of being the default and recommended way of interacting with a database this allows us to make our application much more concurrent and distributed than if we were to hand roll our SQL queries by hand. Phoenix also by default gives us a great logging manager and dashboard to oversee the subprocesses in the system.

# **STAKEHOLDERS AND CONCERNS**

Stakeholders are individuals, groups or organizations that have an interest in the Adrenaline Bikes ERP system. Their areas of interest have an impact on the architecture of the system being developed. For this project, stakeholders are separated in different groups: users, software provider organisation, investors, customer support and material providers.

## **S**takeholders and concerns

2.1.1 Users

1. Client

Clients of Adrenaline Bikes ERP are bike vendors who use the ERP system to manage their business processes.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the information secure? Is the system protected from unauthorized access and leaks of data? | Security |
| Is the information accurate? This includes information regarding orders of parts, client orders, sales, etc. | Correctness |
| Is the system reliable? Are there performance issues? | Performance |
| Is the system easy to use? | Usability |

1. Employee / Administrator

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the information secure? Is the system protected from unauthorized access and leaks of data? | Security |
| Is the information accurate? This includes information regarding orders of parts, client orders, sales, etc. Is the user experiencing bugs? | Correctness |

2.1.2 Negative Users

1. Hackers

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the information secure? Is the system protected from unauthorized access and leaks of data? | Security |

2.1.3 Software Provider Organisation

1. Product Owner

The project owner is responsible for maximizing the value of the product being developed and is responsible for approving and making changes to the requirements during the project lifecycle

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the project on schedule and on budget? | Cost and Schedule |
| Does the system have all the necessary requirements? | Correctness |
| Which methodology should be used in order to maximise the value of the product? | Efficiency |
| Is it easy to maintain the system? | Maintainability |
| Is it possible to scale the system? | Scalability |

1. Software Architect

The software architect is responsible for providing initial models and designs of the system to be built.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Can the different features of the system aptly interact together? | Interoperability |
| Is it easy to maintain the system? Is the technology outdated? Can it be upgraded? Is this the best architecture for a system of the sort? | Maintainability |
| Can the system be adapted to different environments or uses? | Portability |
| Can the system inherit a design from another system? | Efficiency |
| Is it possible to scale the system? | Scalability |

1. Developer

The developers are responsible for developing the features that will make up the system.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Can the different features of the system aptly interact together? | Interoperability |
| Is it easy to maintain the system? If something needs to be changed, can it be done easily? Is the codebase clear with appropriate comments? | Maintainability |
| Can a feature inherit a design from another feature/system? | Efficiency |
| Is it possible to scale the system? | Scalability |
| Is the system performant? | Performance |
| Is the code easy to test? | Testability |

1. Tester

The testers are responsible for writing tests for the software to ensure that the software developer works as expected in all cases.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the code easy to test? | Testability |
| Is the test coverage good? Is there high confidence in the robustness of the code? | Correctness |

2.1.4 Investors

1. Trustee

Trustees are responsible for holding and administering property or assets and have a fiduciary responsibility to carry out the purposes of the trust in the best interest of the beneficiaries.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the project on schedule and on budget? | Cost and Schedule |
| Is the information secure? Is the system protected from unauthorized access and leaks of data? | Security |
| Is the system marketable? Are financial goals being met? | Marketability |

2.1.5 Customer Support

1. Technical support + Customer Service Representation

They are responsible for providing support to people with complaints about the system or people experiencing technical problems.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the system easy to use? Is the overall system of good quality? | Usability |
| Is the system working as expected? Is the user experiencing bugs? | Correctness |

2.1.6 Manufacturing companies

1. Bike equipment production companies

Equipment manufacturing companies are responsible for making the bike parts or bikes that will be sold by companies using the system.

| **Concern** | **Quality Attribute** |
| --- | --- |
| Is the system accurate? Are sales/inventory/orders accurately managed? Is the system working as expected? Is the user experiencing bugs? | Correctness |

## Concern-stakeholder traceability

|  | Client | Employee | Hacker | Product Owner | Architect | Developer | Tester | Trustee | Customer Support | Manufacturing companies |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Correctness | X | X |  | X |  |  | X |  | X | X |
| Performance | X |  |  |  |  | X |  |  |  |  |
| Usability | X |  |  |  |  |  |  |  | X |  |
| Cost and Schedule |  |  |  | X |  |  |  | X |  |  |
| Efficiency |  |  |  | X | X | X |  |  |  |  |
| Maintainability |  |  |  | X | X | X |  |  |  |  |
| Scalability |  |  |  | X | X | X |  |  |  |  |
| Portability |  |  |  |  | X |  |  |  |  |  |
| Interoperability |  |  |  |  | X | X |  |  |  |  |
| Testability |  |  |  |  |  | X | X |  |  |  |
| Marketability |  |  |  |  |  |  |  | X |  |  |
| Security | X | X |  |  |  |  |  | X |  |  |

*THIS PAGE IS INTENTIONALLY BLANK*

# REQUIREMENTS

In this section, we focus on the requirements that have to do with the architecture, since these requirements have impact on the architectural design decisions. The requirements chosen have a measurable effect on a computer system’s architecture. Below is the list of requirements for the Adrenaline Bike ERP system.

## Non-functional Requirements

**Requirement 1: The Adrenaline Bike ERP system shall accommodate to all screen resolutions.**

* Quality Attribute: Usability
* Sub-Quality Attribute: Accessibility
* Subject: Adrenaline Bike ERP
* Action: Being Accessed
* Constraints of Actions: Accessible on all screens resolution and available anywhere

**Requirement 2: The back-end and front-end of Adrenaline Bike ERPshall be independent.**

* Quality Attribute: Maintainability
* Sub-Quality Attribute: Testability, Modularity, Reusability
* Subject: Adrenaline Bike ERP
* Action: Develop back-end and front-end
* Constraints of Actions: Independent of each other

**Requirement 3: Users must register and login to use Adrenaline Bike ERP.**

* Quality Attribute: Security
* Sub-Quality Attribute: Authenticity
* Subject: User
* Action: Register and Login
* Constraints of Actions: Valid users only

**Requirement 4: Adrenaline Bike ERP shall respond within 5 seconds.**

* Quality Attribute: Performability
* Sub-Quality Attribute: Time Behaviour
* Subject: Adrenaline Bike ERP
* Action: Response
* Constraints of Actions: In 5 seconds

**Requirement 5: Adrenaline Bike ERP shall have 95 percent up-time.**

* Quality Attribute: Reliability
* Sub-Quality Attribute: Availability
* Subject: Adrenaline Bike ERP
* Action: Available
* Constraints of Actions: Availability more the 95 percent

**Requirement 6: No more than 1 per 1000 system action should result in a failure such that the application needs a restart.**

* Quality Attribute: Reliability
* Sub-Quality Attribute: Maturity
* Subject: Adrenaline Bike ERP
* Action: Availability
* Constraints of Actions: 999 success actions per 1000 actions

**Requirement 7: In the event of a breakdown, the system will allow a restart period of less than or equal to 120 seconds.**

* Quality Attribute: Reliability
* Sub-Quality Attribute: Recoverability
* Subject: Adrenaline Bike ERP
* Action: Restart
* Constraints of Actions: Below 120 seconds

**Requirement 8: At least 10 users can access the application concurrently.**

* Quality Attribute: Performability
* Sub-Quality Attribute: Capacity
* Subject: Adrenaline Bike ERP
* Action: Accessing the system
* Constraints of Actions: At least 10 users

## Functional Requirements

**Requirement 1: Allow users to register to the Adrenaline Bike ERP**

* Quality Attribute: Usability
* Sub-Quality Attribute: Accessibility
* Subject: Adrenaline Bike ERP
* Action: Register
* Constraints of Actions: Allow valid users

**Requirement 2: Allow registered users to login to the Adrenaline Bike ERP**

* Quality Attribute: Security
* Sub-Quality Attribute: Authenticity
* Subject: Adrenaline Bike ERP
* Action: Login
* Constraints of Actions: Allow valid users

**Requirement 3: A user can try to login 3 times before getting a time penalty.**

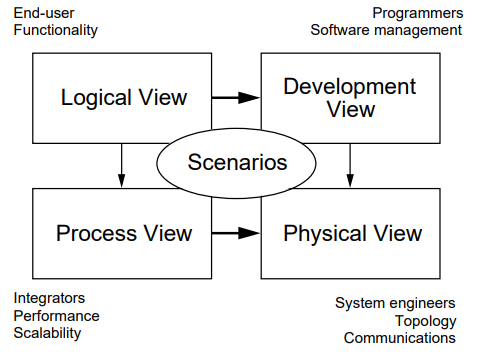
* Quality Attribute: Security
* Sub-Quality Attribute: Authenticity
* Subject: User
* Action: Login
* Constraints of Actions: Allow 3 attempts

**Requirement 4: Allow a registered user to recover a lost password.**

* Quality Attribute: Usability
* Sub-Quality Attribute: Operability
* Subject: Adrenaline Bike ERP
* Action: Recover password
* Constraints of Actions: Valid user

# **VIEWPOINTS** AND VIEWS

The 4+1 model was used during the process of designing the Adrenaline Bike ERP. This mode is used to describe the system from 5 different views as shown in the following figure (Kruchten).

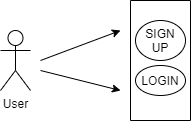


## Scenario view

“User” has different types such as Client, Employee and Administrator. Depending on what user type the user is, they will have different views in Adrenaline Bike ERP.

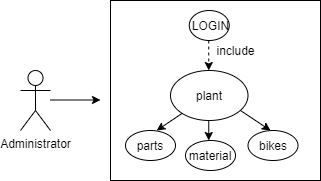
**Sign Up and Login**

User’s must be able to login if they have an existing account and sign up if they do. They are required to submit an email and a password.



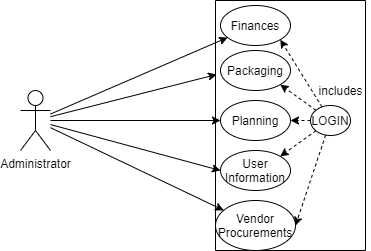
**Production**

Administrators, more precisely manufacturers, have access to all the inventory of the different plant locations. They can see what and how many materials, parts, built bikes and bikes being built there are.



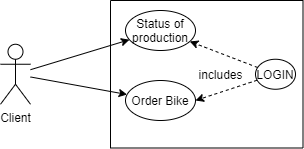
**Admin View**

Administrators have access to the ERP financials, packaging, planning , user information and vendor procurements



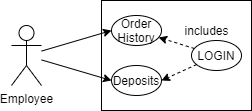
**Client View**

Clients have a specific dashboard that allows them to order bikes and see their status order.



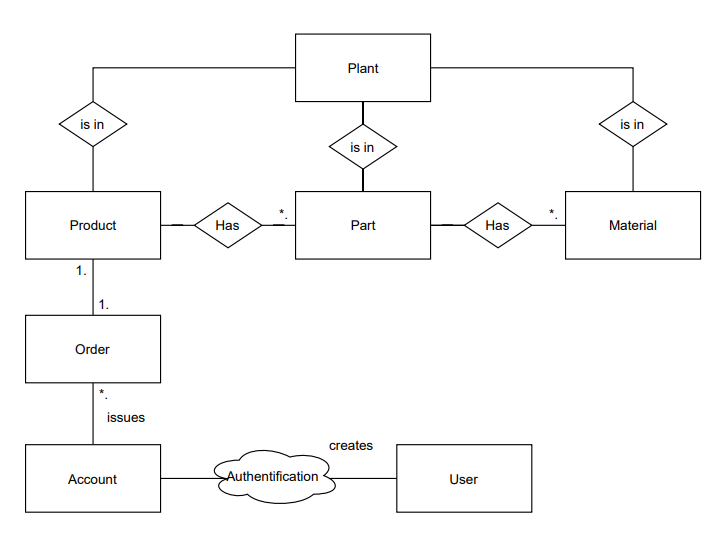
**Employee View**

Employees have access to the different items such as order history and deposits.



## Logical View

The logical view mainly describes architecture of the system required in order to meet functional requirements. This view can be represented using class and state diagrams. These diagrams will illustrate the inheritance and decomposition of the system’s architecture. The following class diagram depicts the structure of the production:

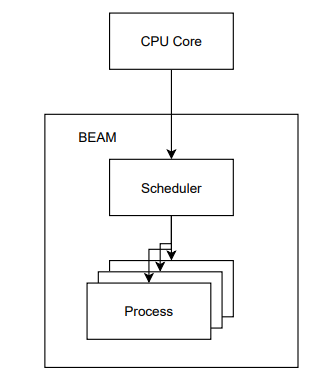


From this diagram it can be understood that products, parts and materials are stored in a plant. Additionally, a user can create an account when authenticated, issue an order and request a product to be assembled. That product will use parts and materials in a specific plant and, when complete, the product will also be stored in that plant.

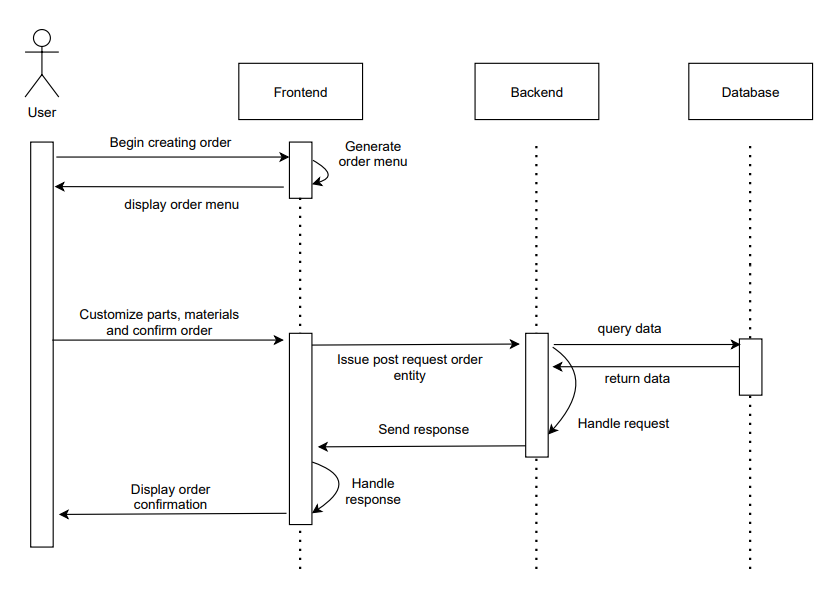
## Process View

The process view addresses non functional requirements such as availability, scalability, concurrence, fault-tolerance and more. Elixir is a language built on Bjorn and Bogdan’s Erlang Abstract Machine (BEAM). Both Elixir and Erlang use BEAM in order to implement the code into an Erlang Abstract Format and eventually into bytecode. Erlang was designed in order to satisfy systems which require high concurrency, availability and fault-tolerance such as telecoms.

For the Adrenaline Bikes ERP, this means that these non functional requirements can be met, since Erlang allows for a large amount of processes to be executed efficiently using schedulers as depicted in the following figure.



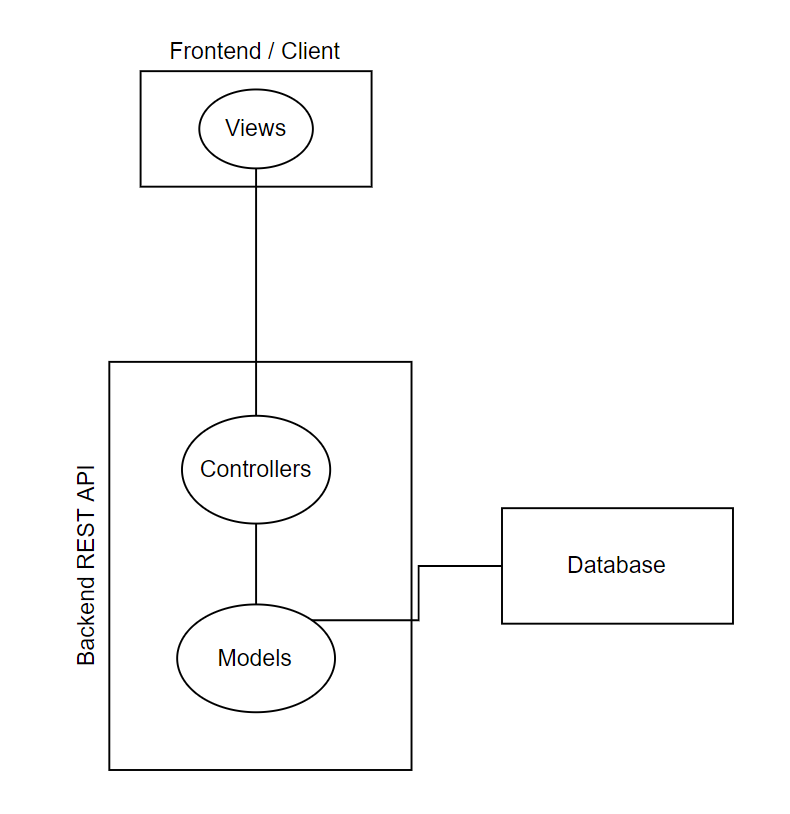
Erlang executes all tasks as a process, which means there is a large number of processes executed during the runtime of the Adrenaline Bikes ERP. Here is a high level view of the user ordering process:



## Development View

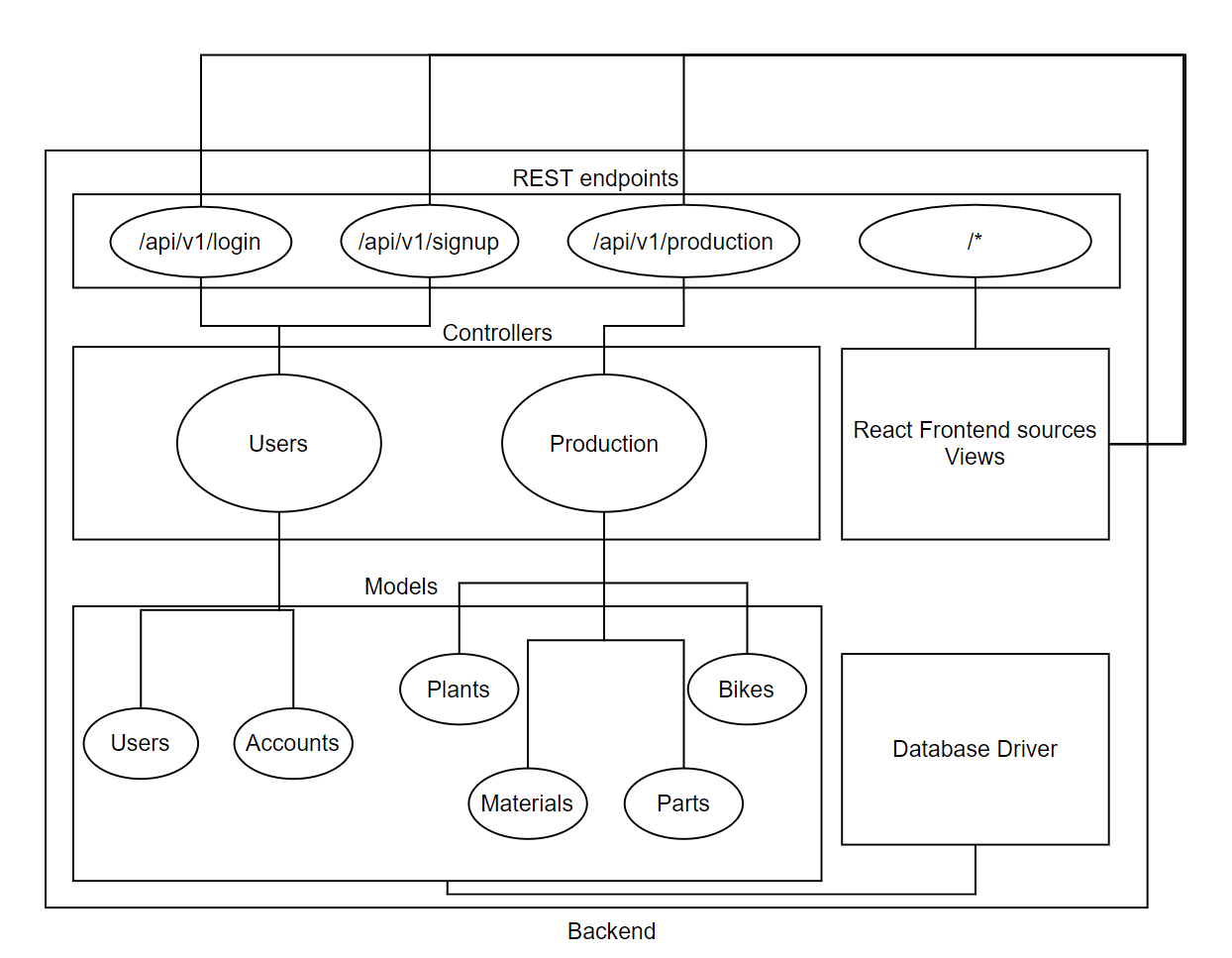
The development view concentrates on how the software is built, at its core. It focuses on how the source code interacts with its different elements and which parts talk to each other.

Here’s what Adrenaline Bike ERP looks like at a high-level. We use the MVC model: A React frontend, which does HTTPS requests to our Phoenix/Elixir backend, and finally our models (which map to a PostgreSQL database) return/modify data by the controllers and finally update the views (React frontend):



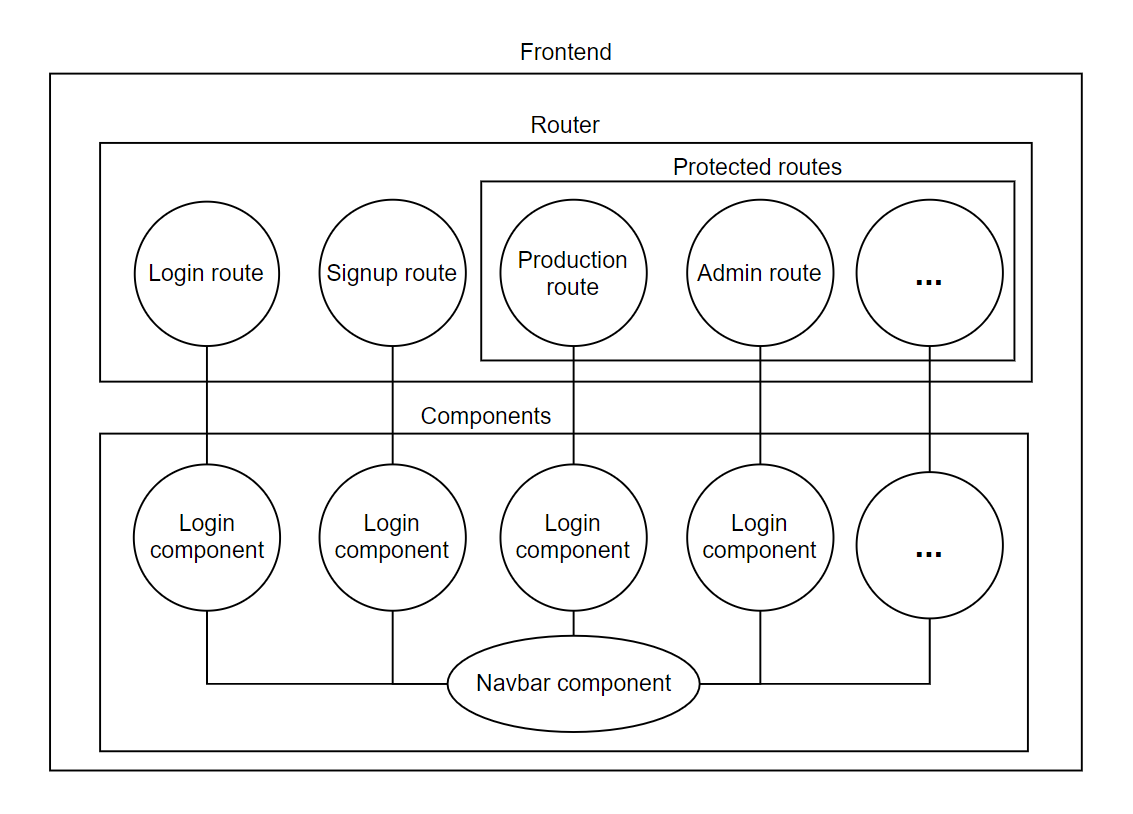
Adrenaline Bike ERP high-level development overview

While the state of our backend is not near being final (this is being written for sprint 1), here is a more in-depth breakdown of our backend and frontend codebase. In the backend, controllers are attached to different HTTPS endpoints, and they will take care of the necessary business logic to return or update data when a view asks it to. They are connected to the endpoints at one end, and to their relevant models at the other. This makes for code that has good context and only serves one purpose, which is the right way of doing it for a functional programming language such as Elixir. Here’s the state of our backend at sprint 1:



Domain Model Diagram of the entire system as of Sprint 1

Our frontend, built with React, has a built-in router for it to act as a single page webapp. Therefore, different routes, and some of which are protected, map to different components that are rendered on-screen.

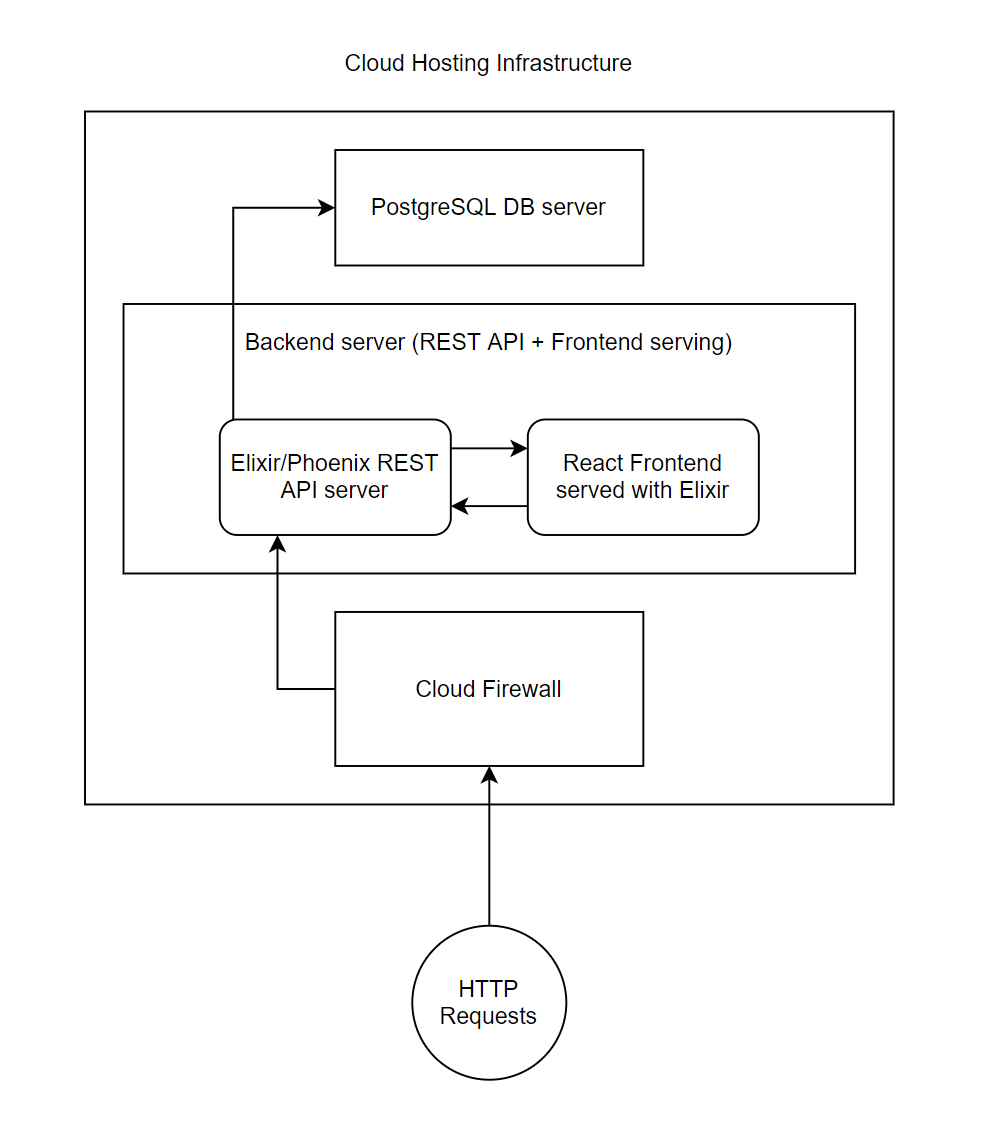


Development view of the frontend

## Physical View

Physical view concentrates on how the software will physically be deployed to be processed on different machines/nodes/clusters. It shows the interactions and connections between the different physical systems that are hosting (processing) the software that enables the ERP system.

Modern software development practices employ the use of the cloud. The cloud is usually a reference to virtual machines, servers, that are operated and maintained elsewhere in the world by a third-party. This third-party, be it AWS, Microsoft Azure or GCP, takes care of the physical devices that host these VMs, their firewall security, their underlying updates and their resource allocation. This allows developers and admins to concentrate on writing good code and letting the hardware maintenance to others. It is good practice to review all networking, firewall, updating, OS and security settings before deploying code to the cloud, but once that’s done, iterating and deploying new code is easy and encouraged to keep software in good shape and secure. DevOps and site admins also benefit from great features such as easy CI/CD and load-balancing, all built into the cloud’s infrastructure.



Adrenaline Bike ERP physical view

# REFERENCES

[1] Kruchten, Philippe. “Architectural Blueprints—The “4+1” View Model of Software Architecture.” *The University of British Columbia*, IEEE Software 12, 1995, https://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf. Accessed 02 02 2021.