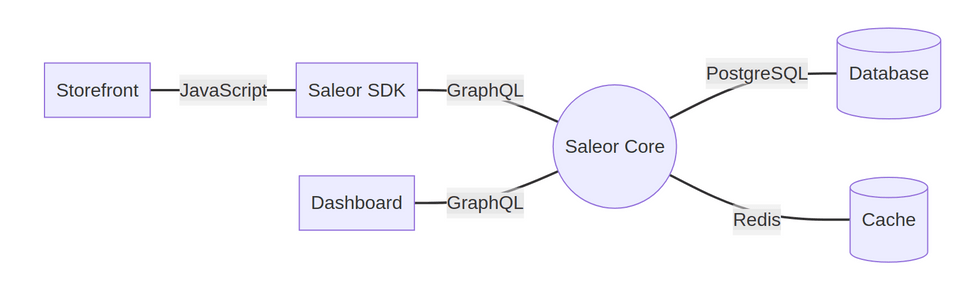
**Saleor Architecture**

**Saleor consists of three distinctive components:**

1. **First is the Saleor Core, the backend server that exposes the GraphQL API The core is written in Python and does not have a user interface. It maintains its state in a PostgreSQL database and caches some information in Redis where available.**
2. **Then, there's the Saleor Dashboard, which implements the user interface that staff members can use to operate a store. The dashboard is a React application that runs in the browser and talks to the core server. It's a static website, so it does not have any backend code.**
3. **Lastly, there's the Next.js Storefront, an example storefront implemented in React with Next.js. You can customize its code to suit your needs or build a custom storefront using the underlying Saleor SDK.**

**All three components communicate using GraphQL over HTTPS.**

****

**Microservices**

 Headless commerce is one version of a microservices approach because it involves decoupling aspects of the website and allowing them to work independently. However, with headless, only some of the parts of the system are decoupled (the front-end from the back-end).

Modular Folders

Each feature has a separate folder. It follows a 12-factor code structure.

It is divided as

Core, Features. Modules, Files.

-- modules  
|-- home  
|-- components  
|-- pages  
| |-- home  
| |-- home.component.ts|html|scss|spec  
|  
|-- home-routing.module.py  
|-- home.module.py

**Middleware**

Middleware is a framework of hooks into Django’s request/response processing. It’s a light, low-level “plugin” system for globally altering Django’s input or output. Each middleware component is responsible for doing some specific function.

MIDDLEWARE = [

"django.middleware.security.SecurityMiddleware",

"django.middleware.common.CommonMiddleware",

"saleor.core.middleware.request\_time",

"saleor.core.middleware.google\_analytics",

"saleor.core.middleware.jwt\_refresh\_token\_middleware",

]

**Code quality**

The project has configured Prettier and ESLint.

ESLint is a tool for identifying and reporting on patterns found in ECMAScript/JavaScript code, with the goal of making code more consistent and avoiding bugs. In many ways, it is similar to JSLint and JSHint with a few exceptions: ESLint uses Espree for JavaScript parsing.

**Headless**

Headless commerce, as an approach, decouples the front end and back end of businesses’ web stores by putting an application programming interface (API) between them that manages the exchange of data, creating an opportunity to test and experiment within both environments (independent of each other). Although with headless commerce the front end and back end are decoupled, this does not mean that they are completely disconnected; they simply can be experimented with separately. They remain connected through web services or API calls, which keeps the data exchange between the decoupled systems intact, even as the organization experiments with both environments.

**How headless eCommerce works**

Just as with a headless CMS, a headless eCommerce system works by passing requests between the presentation and application layers through web services or application programming interface (API) calls.

For instance, when the user clicks a “Buy Now” button on their smartphone, the presentation layer of the headless eCommerce system sends an API call to the application layer to process the order. The application layer sends another API call to the application layer to show the customer the status of their order.

**Advantages**

Agility to remain competitive

Flexibility of customizations

Better (and always-evolving) customer experience

**PostgreSQL**

PostgreSQL is an object-relational database management system (ORDBMS) based on **POSTGRES, Version 4.2**, developed at the University of California at Berkeley Computer Science Department. POSTGRES pioneered many concepts that only became available in some commercial database systems much later.

PostgreSQL is an open-source descendant of this original Berkeley code. It supports a large part of the SQL standard and offers many modern features:

* complex queries
* foreign keys
* triggers
* updatable views
* transactional integrity
* multiversion concurrency control

Also, PostgreSQL can be extended by the user in many ways, for example by adding new

* data types
* functions
* operators
* aggregate functions
* index methods
* procedural languages

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.

Facebook's mobile apps have been powered by GraphQL since 2012. GitHub and Coursera are also using it.

**Redis Cache**

what is Redic cache? When it comes to Redis, Redis is short for Remote Dictionary Server. Redis is a caching system that works by temporarily storing information in a key-value data structure.

How Redis cache works is by assigning the original database query as the key and then resulting data as the value. Now, the Redis system can access the resulting database call by using the key that it has stored in its built-in temporary memory.

Next.js

Next.js is a React framework that **gives you building blocks to create web applications**. By framework, we mean Next.js handles the tooling and configuration needed for React, and provides additional structure, features, and optimizations for your application.

Why Next.js for eCommerce?

Next.js’ tooling lends itself well to eCommerce stores in general. It allows developers to build high-performing statically generated product display pages and catalogs.

Overall, Next.js commerce has a rather ambitious goal: to become a common interface across every platform. By abstracting a lot of complexity away from developers, it is one of the most flexible, easy-to-customize JavaScript frameworks for developing web applications.

With the help of a headless CMS or a headless commerce platform as a backend, Next.js to build the frontend and third-party tools like product information management solutions and search engines, you’re all set for success.

**Saleor SDK**

This package contains methods providing Saleor business logic for a storefront and apps. It handles Saleor GraphQL queries and mutations, manages Apollo cache, and provides an internal state to manage popular storefront use cases, like user authentication.

**API vs Webhooks**

When a client API requests data from a server API, it’s calling to see if a certain event has occurred—in other words, whether the server’s data has changed in a way that might be useful to the client. In this process (known as *polling*), the client sends an HTTP request at regular intervals until the server’s API sends the relevant data, which is sometimes called the *payload*.

The client app doesn’t know the state of the server app, so it polls the server’s API for an update—calling over and over until the specific event occurs—but the server will only send the requested data once that information is available. The client app has to keep asking for the update and wait until the relevant event happens.

To set up a webhook, the client gives a unique URL to the server API and specifies which event it wants to know about. Once the webhook is set up, the client no longer needs to poll the server; the server will automatically send the relevant payload to the client’s webhook URL when the specified event occurs.

Webhooks are often referred to as *reverse APIs* or *push APIs*, because they put the responsibility of communication on the server, rather than the client. Instead of the client sending HTTP requests—asking for data until the server responds—the server sends the client a single HTTP POST request as soon as the data is available. Despite their nicknames, webhooks are not APIs; they work together. An application must have an API to use a webhook.

1. manage.py

This file is used as a command-line utility for our projects. We will use this file for debugging, deploying, and running our web applications.

1. \_\_init.py\_

This is an empty file as you can see below in the image. The function of this file is to tell the Python interpreter that this directory is a package and involvement of this \_\_init.py\_ file in it makes it a python project.

2. settings.py

It contains the Django project configuration.

The setting.py is the most important file, and it is used for adding all the applications and middleware applications. This is the main setting file of the Django project.

3. urls.py

URL is a universal resource locator, it contains all the endpoints that we should have for our website. It is used to provide you the address of the resources (images, webpages, websites, etc) that are present out there on the internet.

4. wsgi.py

When you will complete your journey from development to production, the next task is hosting your application. Here you will not be using the Django web server, but the WSGI server will take care of it

WSGI stands for **Web Server Gateway Interface**, it describes the way how servers interact with the applications.

5. asgi.py

ASGI works similar to WSGI but comes with some additional functionality.  ASGI stands for **Asynchronous Server Gateway Interface**. It is now replacing its predecessor WSGI.

**PWA**

module.exports = {

reactStrictMode: true,

};

const withPWA = require("next-pwa");

module.exports = withPWA({

pwa: {

dest: "public",

register: true,

disable: process.env.NODE\_ENV === 'development',

skipWaiting: true,

},

});

Docasaurous

 Docusaurus will help you ship a **beautiful documentation site in no time**.

**Translation**

SRS uses FormatJS as a library for translating the interface.

Translation sources are located in the locale directory. Please note that SRS uses full language code names to differentiate between language dialects (e.g., en-US and en-UK).

**Adding new languages**

Create a new json file in the locale directory

Add a language to the LOCALES constant in the lib/regions.ts file

Extend the importMessages function to import the proper json file based on chosen locale

**Security**

Depending on the content of the personal data submitted by the User such data are processed by us for the performance of a contract or in order to take steps prior to entering into a contract (legal basis: Article 6(1)(b) of the GDPR) or for maintaining contact and relationships with clients or potential clients based on our legitimate interests as data controller (legal basis: Article 6(1)(f) of the GDPR).

The dotenv package is a great way to keep passwords, API keys, and other sensitive data out of your code. It allows you to create environment variables in a . env file instead of putting them in your code.

**Deployement**

Docker is Saleor's preferred deployment method.

Core

To build the image, use docker build:

cd saleor

docker build -t <yourimagename> .

**SEO optimisation**

Take care of organic traffic by creating SEO-optimised content for products, collections, and categories.

**Multi-warehouse inventory**

Manage multiple locations seamlessly by tracking stock levels, editing multiple products at once and checking items that aren’t available anymore.

**CSV import and export**

Saleor makes it possible to import and export any product data to Google and Excel.

**Multi-language**

Saleor already offers more than 30 languages.

**Translations module**

Translate dynamic content in the dashboard to create different language versions if you need them.

**Localised data**

Saleor takes advantage of geolocation – it checks your location and matches it with the location of your customers. This way, you can sell both locally and globally.

**Global support**

Whether you are in Europe or the US, Saleor supports you in building better shopping experiences.

Poetry file

 the poetry. lock file **prevents you from automatically getting the latest versions of your dependencies**. To update to the latest versions, use the update command. This will fetch the latest matching versions (according to your pyproject. toml file) and update the lock file with the new versions

Cypress is a purely JavaScript-based front-end testing tool built for the modern web. It aims to address the pain points developers or QA engineers face while testing an application. Cypress is a more developer-friendly tool that uses a unique DOM manipulation technique and operates directly in the browser

What is env CMD?

env-cmd ^10.1. 0 npm package: A simple node program for executing commands using an environment from an env file.

What is husky in JS?

Husky is a JavaScript package that allows you to run some code during various parts of your git workflow. Husky leverages git hooks to allow you to hook into various git events such as pre-commit and pre-push.

What is lint-staged?

lint-staged just changes your code and make it linting (It runs before commit by husky). After changed, you need add it again to update git index. And your changes will be effect in your commit.

Prettier enforces a consistent code style (i.e. code formatting that won't affect the AST) across your entire codebase because it disregards the original styling\* by parsing it away and re-printing the parsed AST with its own rules that take the maximum line length into account, wrapping code when necessary.

Heroku is a platform as a service (PaaS) that enables developers to build, run, and operate applications entirely in the cloud.

**Features**

* **Headless / API first**: Build mobile apps, custom storefronts, POS, automation, etc
* **Extensible**: Build anything with webhooks, apps, metadata, and attributes
* **GraphQL API**: Get many resources in a single request and [more](https://graphql.org/)
* **Multichannel**: Per channel control of pricing, currencies, stock, product, and more
* **Enterprise ready**: Secure, scalable, and stable. Battle-tested by big brands
* **CMS**: Content is king, that's why we have a kingdom built-in
* **Dashboard**: User friendly, fast, and productive. (Decoupled project [repo](https://github.com/saleor/saleor-dashboard) )
* **Global by design** Multi-currency, multi-language, multi-warehouse, tutti multi!
* **Orders**: A comprehensive system for orders, dispatch, and refunds
* **Cart**: Advanced payment and tax options, with full control over discounts and promotions
* **Payments**: Flexible API architecture allows integration of any payment method
* **SEO**: Packed with features that get stores to a wider audience
* **Cloud**: Optimized for deployments using Docker

All three components communicate using GraphQL over HTTPS.

* **Blazing speed**– page speed and performance are crucial Google ranking factors and reasons why customers choose (or not) a particular online store. Saleor helps you delight users with a shopping experience that loads in a blink of an eye.
* **SEO efficiency** – the front-end layer in Saleor shops is static, and Google favours static pages as they are fast, light and easy to scan for robots.
* **Accessibility on any device** – your online store will work seamlessly on all devices and screen sizes.
* **Rich user experience** – having a good offer isn’t good enough, and creating a custom user experience is one of the things that can give you a competitive advantage. Experiment with your design without compromising the page load speed.
* **Instant changes** – once the developer makes changes, these will be visible to the customers.
* **Safety** – on contrary to dynamic pages, static ones (the front-end layer of Sailor’s webshop are static) have no direct connection to the database, dependencies, user data, or other sensitive information. Your customers’ data is safe.