Team Project

CII4SE003CI c2

Table of Contents

Teams

Scrum Master: Aland Al-Jaf 52879

Team Lead: Emmanuel Amaechi 52851

TEAM 1: UML DESIGN

1. Aland Al-Jaf 52879
2. Emmanuel Amaechi 52851
3. Colin Onions 52935
4. Eden Mengesha 52844
5. Kanat Kerimbaev 52929

TEAM 2: DATABASE

1. Aland Al-Jaf 52879
2. Xuan yang 49877
3. Ujjwal Rastogi 52911
4. Sofonias Aberra 52877
5. Yonas Lemma 52962

TEAM 3. USER INTERFACE

1. Emmanuel Amaechi 52851
2. Aland Al-Jaf 52879
3.
4.
5.

TEAM 4: SECURITY

1. Maksym Bordun 57354
2. Prayer Mafaune 52991
3.
4.
5.

Project Introduction

Project name: EnRoute

Application Goal

The plan is to make day-to-day grocery shopping more efficient for those who do not have time to go grocery shopping or do regular shopping while at work.

What is the solution?

Many people who live alone or do not have enough time to do their daily/weekly grocery shopping, or those who are unable to do their shopping due to disability issues. The software allows anyone to Buy/Sell goods without having to leave their homes and have them delivered to their desired address. Also, during this COVID-19 era, minimal contact helps protect customers, employees, and the public at large from the spread of the virus.

What Does the Application Do?

Buyers can purchase goods and products from markets that have their products listed on the application. There are three types of clients. The first are individuals that create a market on the application and list their goods for sale to others. The second are those who buy goods from sellers. Lastly, the third are those that deliver goods from the seller's warehouse/store to the buyer's address.

Functionalities (Entities)

1. User Account
    a. ID
    b. Name
    c. Address
    d. Type
        i. Buyer
        ii. Seller
        iii. Delivery
2. Order
    a. ID
    b. Basket
        i. Add Item
        ii. Remove Item
        iii. Apply Coupon
    c. Save
    d. Go To Checkout
3. Item
    a. ID
    b. Name
    c. Product Date
        i. Production Date
        ii. Expiry Date
    d. Seller ID
4. Checkout
    a. Payment Option
        i. Cash
        ii. Card
    b. Delivery Option
        i. Pick-up
        ii. Delivery
            1. Shipping Address
    c. Place Order
5. Order Shipment
    a. User Seller ID
    b. Status
    c. User Delivery ID
    d. Expected Packaging Date
    e. Comment
6. Delivery Shipment

    a. User Delivery ID
    b. Status
    c. User Seller ID
    d. User Buyer ID
    e. Expected Shipment Date
7. Trusted Delivery User
    a. User Delivery ID
    b. User Seller ID
    c. Contract Starts
    d. Contract Ends
    e. Payment Option
    f. Payment Type
        i. Delivery Based
        ii. Hourly Based
        iii. Monthly Based

UML Diagram + Use Cases

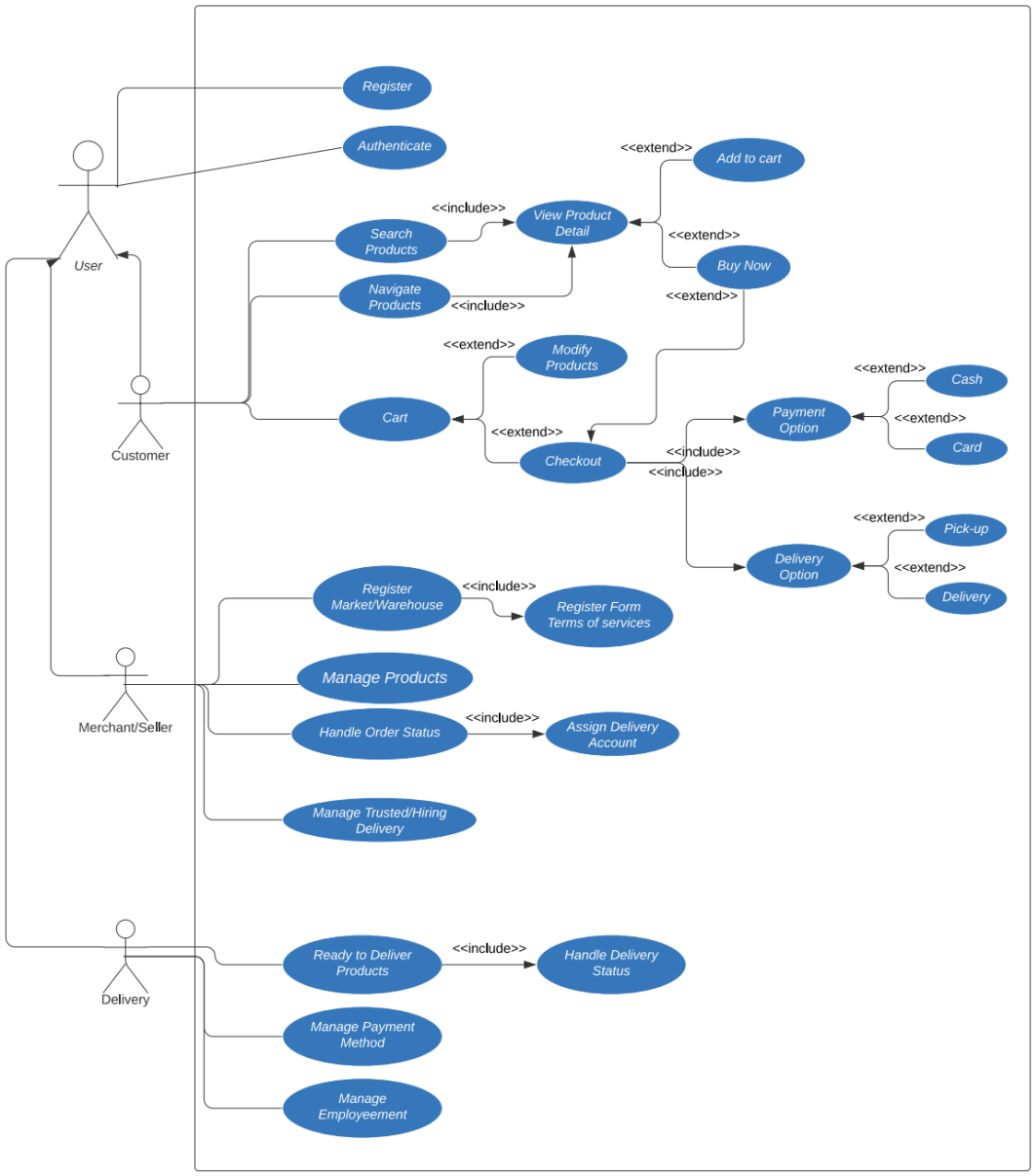
Use Cases

1. Buyer/Seller/Delivery Users
    a. Buyers can buy products
    b. Sellers can register their market/warehouse on the application
    c. Delivery Users can deliver products from market/warehouse location to the buyer address
2. Sellers create a market on the application and add products to it under their market name. Also, they have the option of using trusted delivery accounts or hiring delivery accounts for immediate delivery options when customers place orders in their shop
3. Buyers buy goods from markets/warehouses and add the items to their basket. Then, at the checkout page
    a. Product pick-up or delivery option
    b. Payment option is cash or card
    c. After they place their order successfully, the order is sent to the server for further processes.
4. Delivery Users can be employed by sellers or added to the seller account's trusted list. When a customer selects the delivery option, all delivery accounts will see deliveries on their **ready-to-deliver** page on the application and will be able to deliver the products to the specified address.

UML DESIGN FILE

Link To UML Diagram

EnRoute Use-Case Diagram



Database Design

A well-structured database is needed for such a complex application that requires a lot of queries and recommendations. For this purpose, our team chose a relational database that helps our application to easily query multiple data among the tables and retrieve relational data to the specific query. The primary benefit of the relational database approach is the ability to create meaningful information by joining the tables that allows us to understand the relationships between the data, or how the tables connect. Those features make the relational approach the single most popular query tool in business today.

Relational databases have several advantages compared to other database formats

Flexibility:
Data Definition Language (DDL) is a built-in language in SQL for creating tables. DDL allows you to add new columns, tables, rename relations, and make other changes while the database is running, and queries are being executed. This enables you to modify the schema or the way you model data on the fly.

Reduced Redundancy:
Relational databases eliminate data redundancies. The information for a single customer appears in one place—a single entry in the customer table. The order table only needs to store a link to the customer table. The practice of separating the data to avoid redundancy is called normalization. Progressional database designers make sure the tables normalize during the design process.

Ease of Backup and Disaster Recovery:
Relational databases are transactional—they guarantee the state of the entire system is consistent at any moment. Most relational databases offer easy to export and import options, making backup and restore trivial. These exports can happen even while the database is running, making restore on failure easy. Modern, cloud-based relational databases can do continuous mirroring, making the loss of data on restore measured in seconds or less. Most cloud-managed services allow you to create Read Replicas. These Read Replicas enable you to store a read-only copy of your data in a cloud data center. Replicas can be promoted to Read/Write instances for disaster recovery as well.

ENTITIES

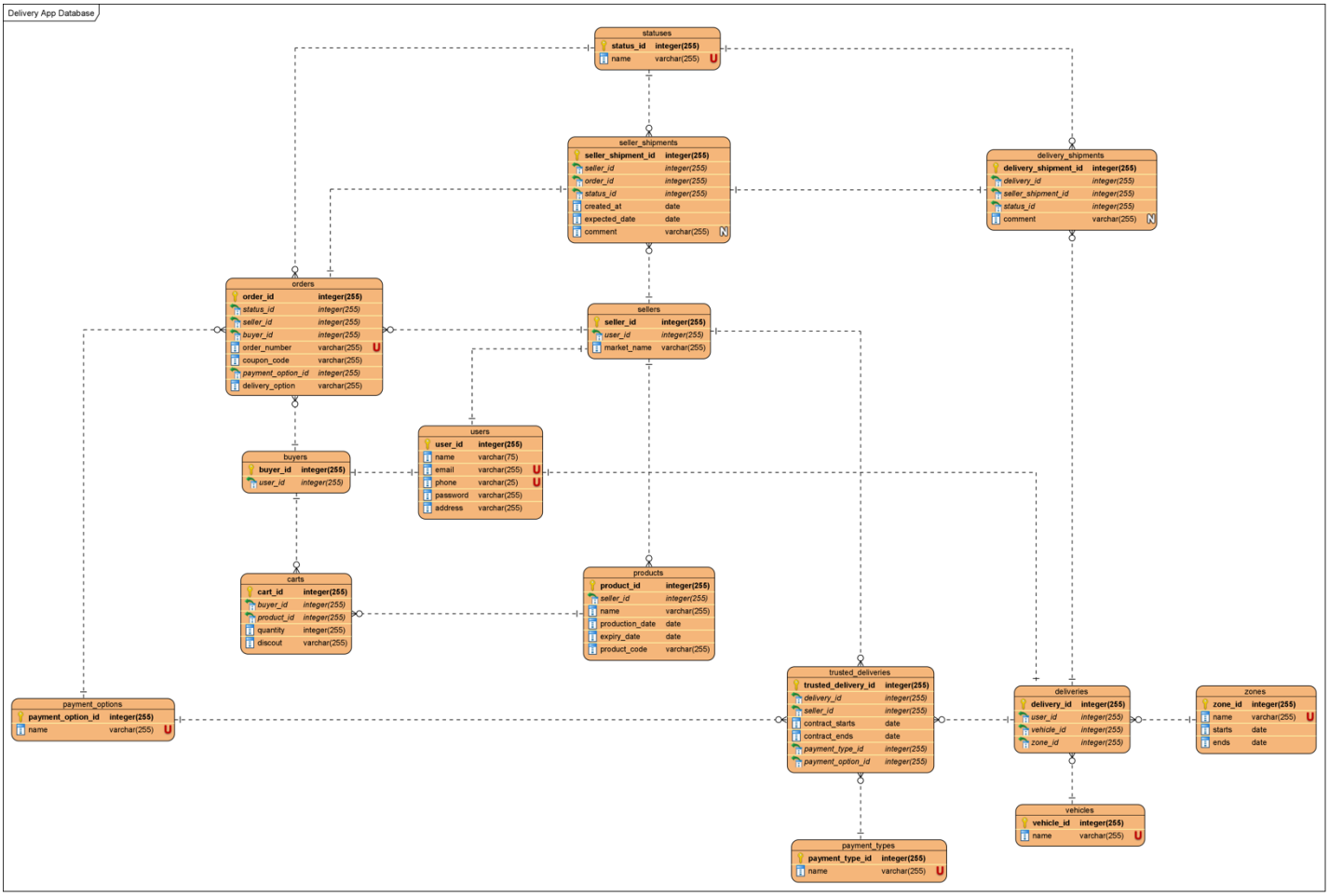
- Users
    - User_id (PK)
    - Name
    - Email (Unique)

    - Phone (Unique)
    - Password
    - Address
- Buyers
    - Seller_id (PK)
    - User_id (FK), (one-to-one)
- Sellers
    - Seller_id (PK)
    - User_id (FK), (one-to-one)
    - Market_name
- Deliveries
    - Delivery_id (PK)
    - User_id (FK), (one-to-one)
    - Vehicle_id (FK), (one-to-one)
    - Zone_id (FK), (one-to-one)
- Orders
    - Order_id (PK)
    - Status_id (FK), (many-to-one)
    - Seller_id (FK), (many-to-one)
    - Buyer_id (FK), (many-to-one)
    - Order_number (Unique)
    - Coupon_code
    - Payment_option_id (FK), (many-to-one)
    - Delivery_option
- Carts
    - Cart_id (PK)
    - Buyer_id (FK), (many-to-one)
    - Product_id (FK), (many-to-one)
    - Quantity
    - Discount
- Products
    - product_id (PK)
    - seller_id (FK), (many-to-one)
    - Name
    - Production_date
    - Expiry_date
    - Product_code
- Statuses
    - Status_id (PK)

    - Name (Unique)
- Payment_options
    - Payment_option_id (PK)
    - Name (Unique)
- Seller_shipments
    - Seller_shipment_id (PK)
    - Seller_id (FK), (one-to-many)
    - Order_id (FK), (one-to-one)
    - Status_id (FK), (one-to-many)
    - Created_at
    - Expected_date
    - Commnet (Nullable)
    -
- Delivery_shipments
    - Delivery_shipment_id (PK)
    - Delivery_id (FK), (one-to-many)
    - Seller_shipment_id (FK), (one-to-one)
    - Status_id (FK), (one-to-many)
    - Commnet (Nullable)
- Vehicles
    - Vehicle_id (PK)
    - Name (Unique)
- Payment_types
    - Payment_type_id (PK)
    - Name (Unique)
- Zones
    - Zone_id (PK)
    - Name (Unique)
    - Starts
    - Ends
- Trusted_deliveries
    - Trusted_delivery_id (PK)
    - delivery_id (FK), (one-to-many)
    - Seller_id (FK), (one-to-many)
    - Contract_starts
    - Contract_ends
    - Payment_type_id (FK), (one-to-many)
    - Payment_option_id (FK), (one-to-many)

DATABASE DESIGN FILE

[Link To PDF Design]



User Interface Design

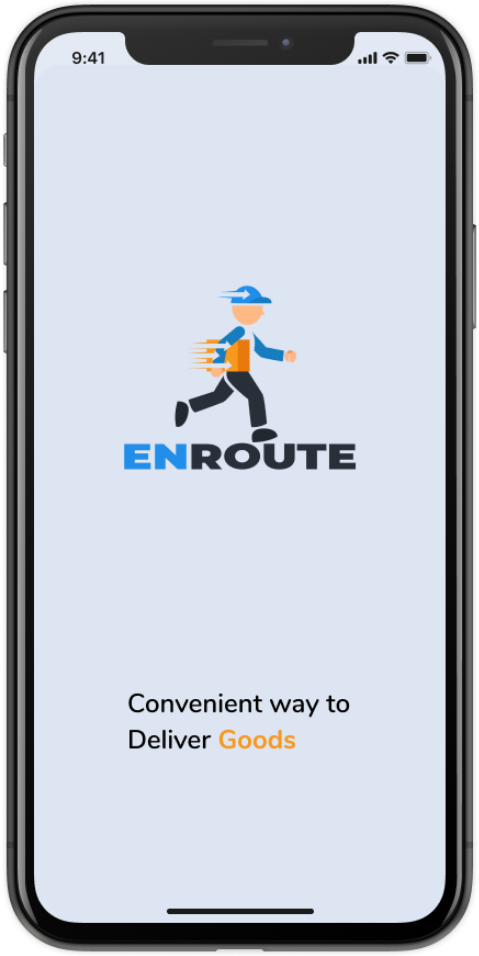
Everyone wants to experience the best technology without spending too much time. The same goes for browsing websites or mobile applications, where the faster and more efficiently a website or mobile application responds, the more successful the outcome. Therefore, we see the design interface for the application as an essential part of the application. We tried to make the interface as simple as possible and implement all the features we could provide through our service.

Application Design Zip File:

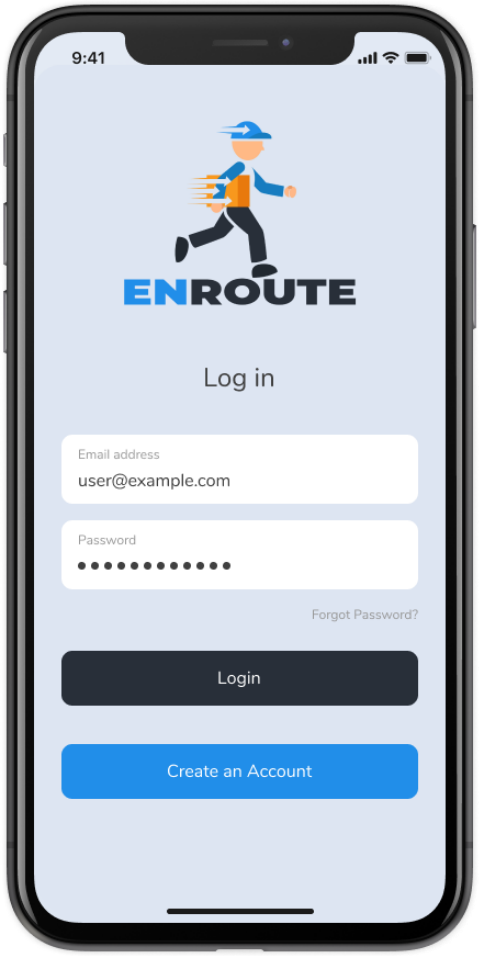
Link To Design Files

Starting Pages:

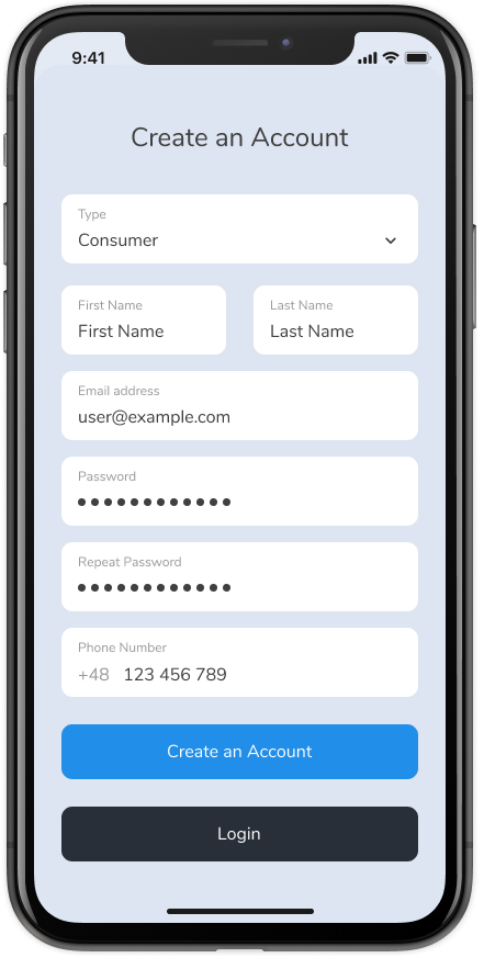
- A flash screen with the App's logo and slogan is displayed upon booting.



- Shortly after the flash screen, users are automatically redirected to a login screen to log into their account. If a person does not currently have an account, they can create one by clicking the "Create an Account" button, which will send them to the Registration form.

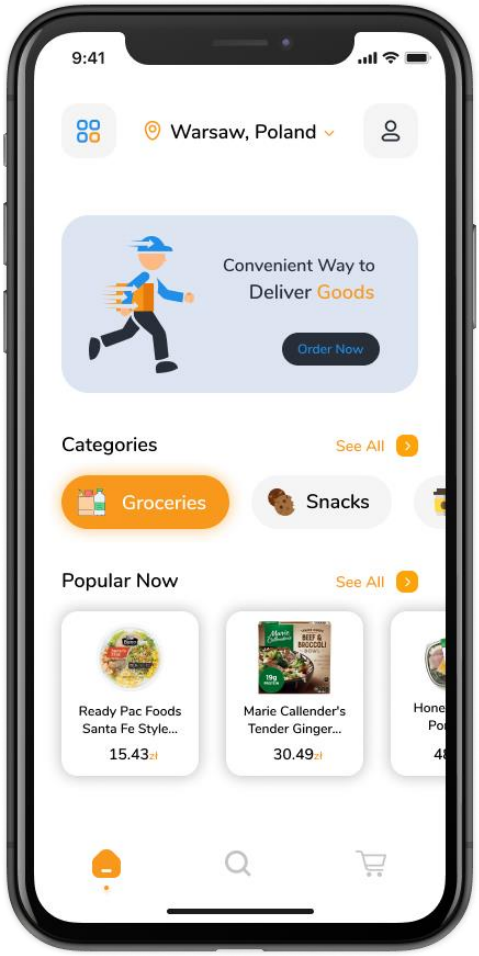


- Users can register as Merchants, Consumers, or Delivery agents on the registration form, which lists the three categories of services offered by EnRoute.



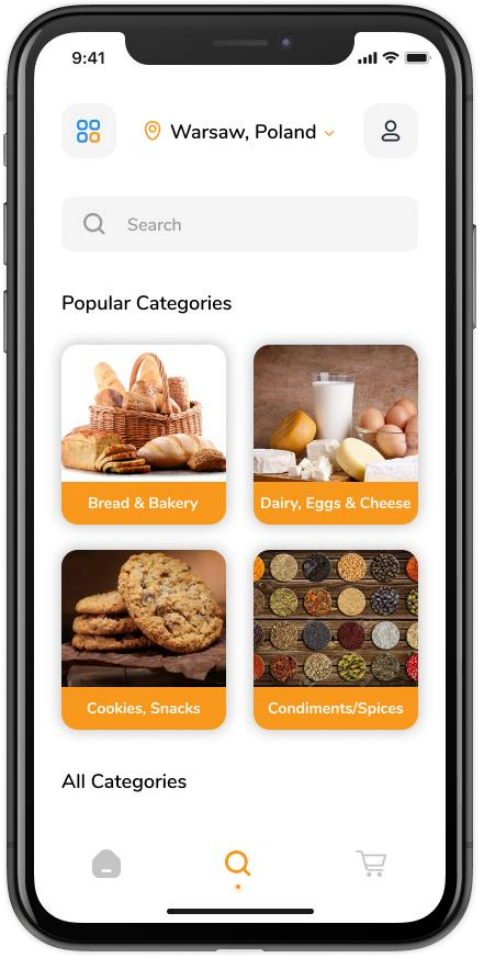
Home Page:

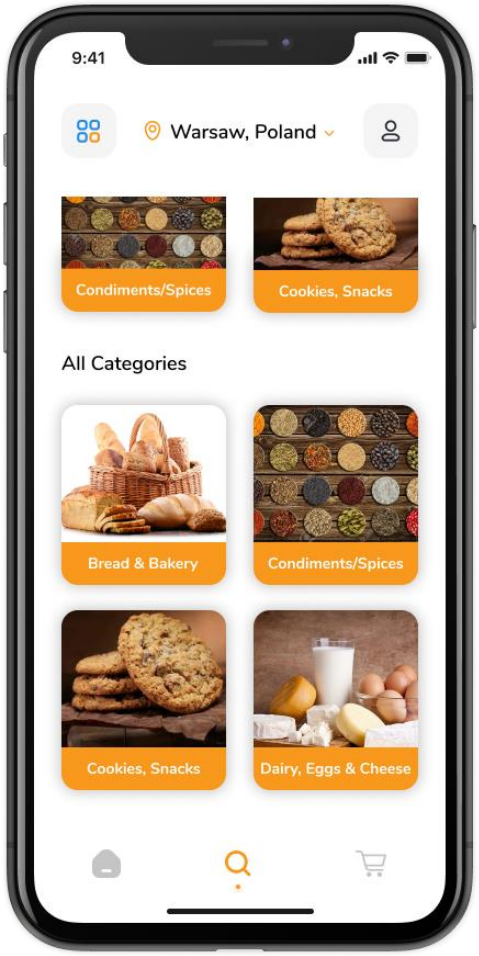
- After a successful sign-in/registration, a user is sent to the Home Page.
- In addition, if a user has initially logged in, subsequently, they will be automatically redirected to the app's home screen after it loads.



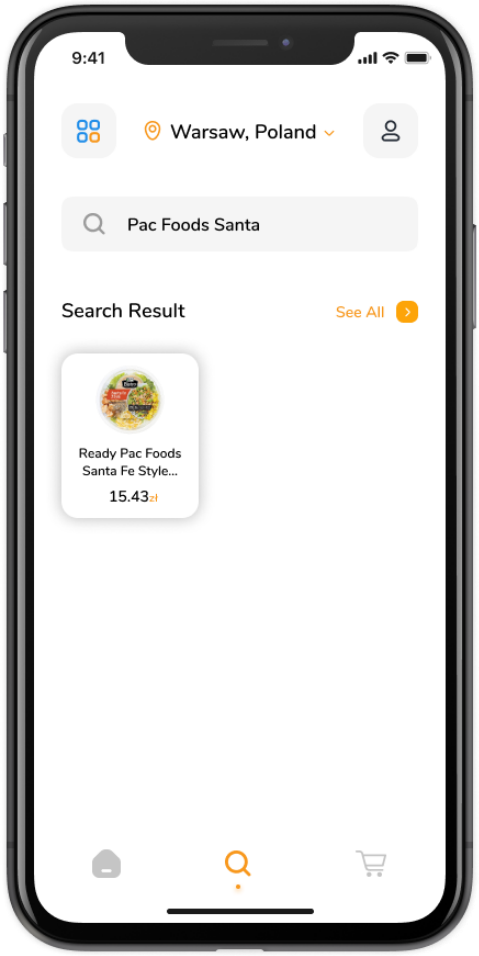
Search Pages:

- The program has a search screen to look for products or browse product categories.
- Users can utilize the search feature to rapidly find a specific product rather than sifting through a large number of options.
- Users can also browse the most popular categories or all the categories offered in our app.


9:41
Warsaw, Poland
Search
Popular Categories
Bread & Bakery
Dairy, Eggs & Cheese
Cookies, Snacks
Condiments/Spices
All Categories



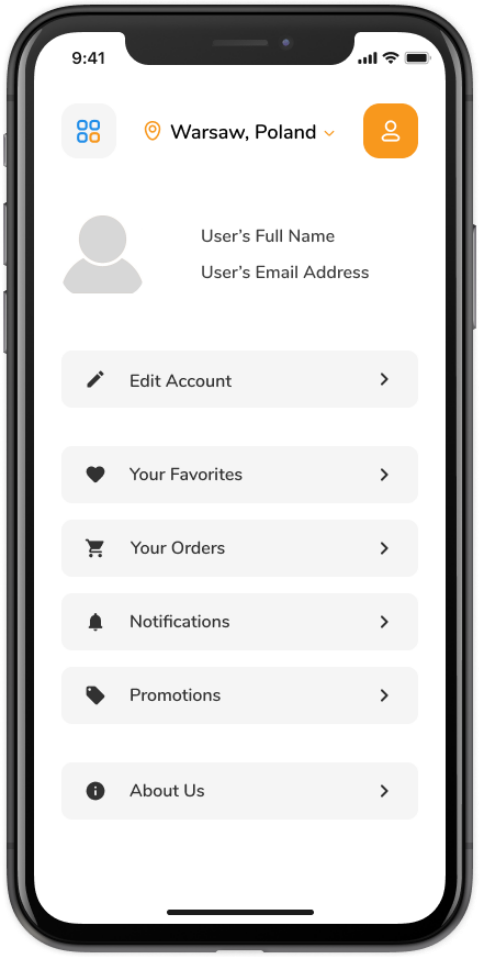
- The design below depicts the look of the page with specific product search results.
- Because there could be more than 10 products related to a given search, we limited the option to keep the user's page from becoming cluttered. As a result, visitors can utilize the "**See All**" option to see all products that match the searched text.

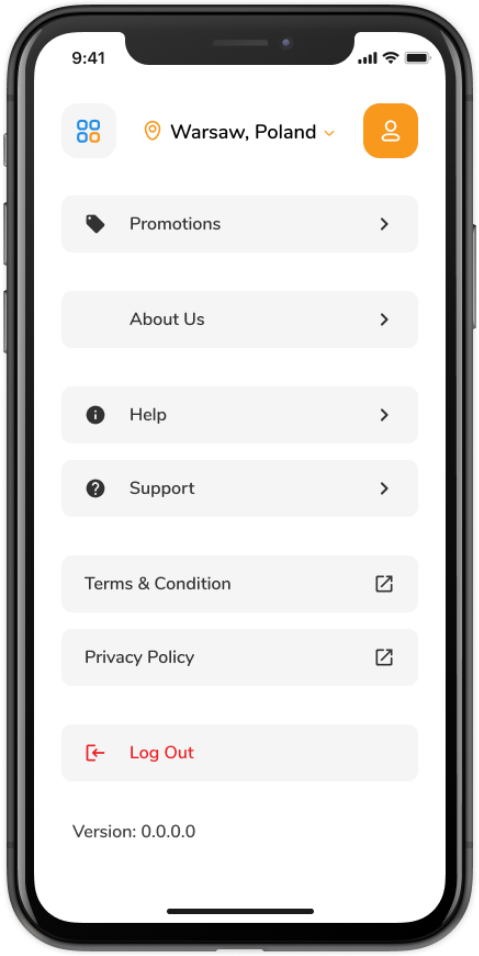


Account Settings Pages:

- By clicking the top-right icon on the app, you may get to the account settings page.
- This page displays all the user's account management and order history.
- The following features are available on this page:
    - Edit Account: ***Look at A.1 for Reference***
    - Your Favorite: This page contains all the products that the user marked as favorite.
    - Your orders: ***Look at A.2 for Reference***
    - Notification: This page contains all the notifications for the purpose of notifying the user for a certain task or action.
    - Promotion: Users can look at all the products that have discounts or find promotions.
    - About Us: ***Look at A.3 for Reference***

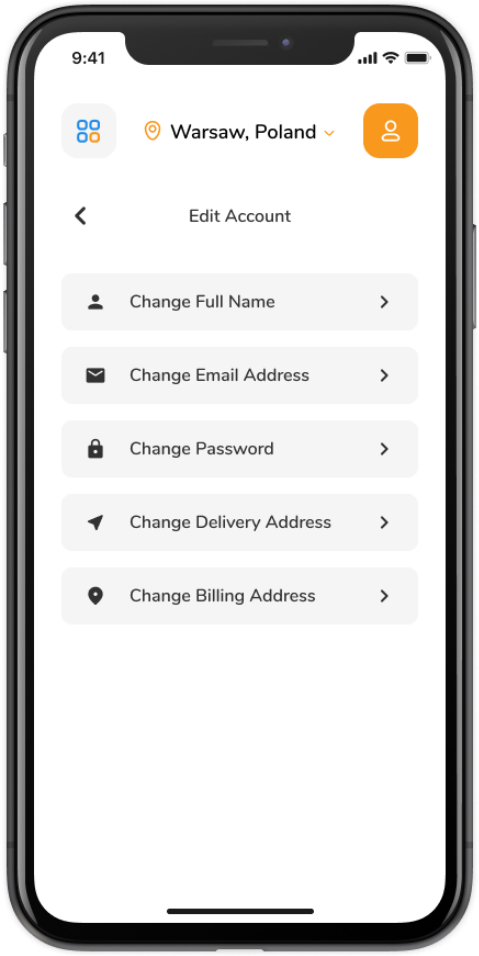
- Help: All the necessary guides for users to use our service or to find answers to their questions are available.
- Support: Users can contact our support team for any technical difficulties.
- Terms & Conditions: Link to our terms of use.
- Privacy Policy: Link to our privacy policy.
- Logout: log out the current user.





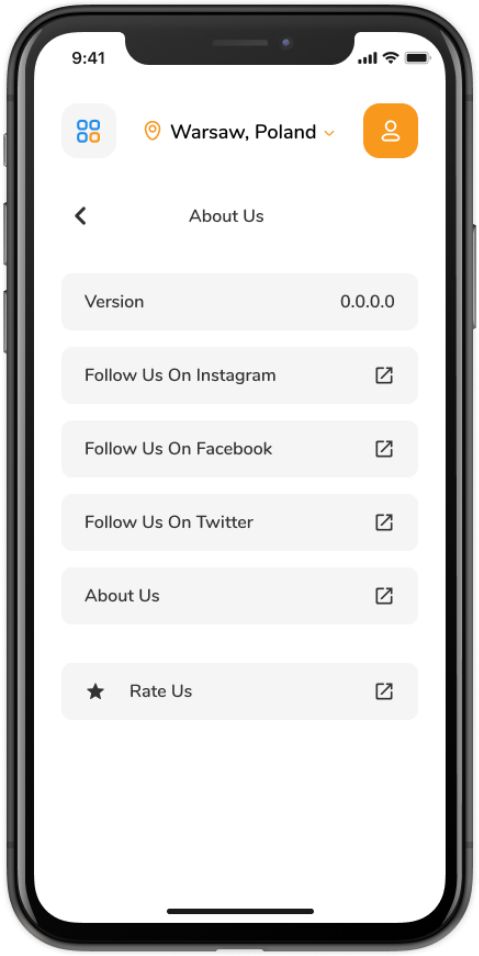
Edit Account Section (Reference A.1):

- Edit Account section allows users to change the following information about their account.
    - Full Name
    - Email Address
    - Password
    - Delivery Address
    - Billing Address



About Section (Reference A.3):

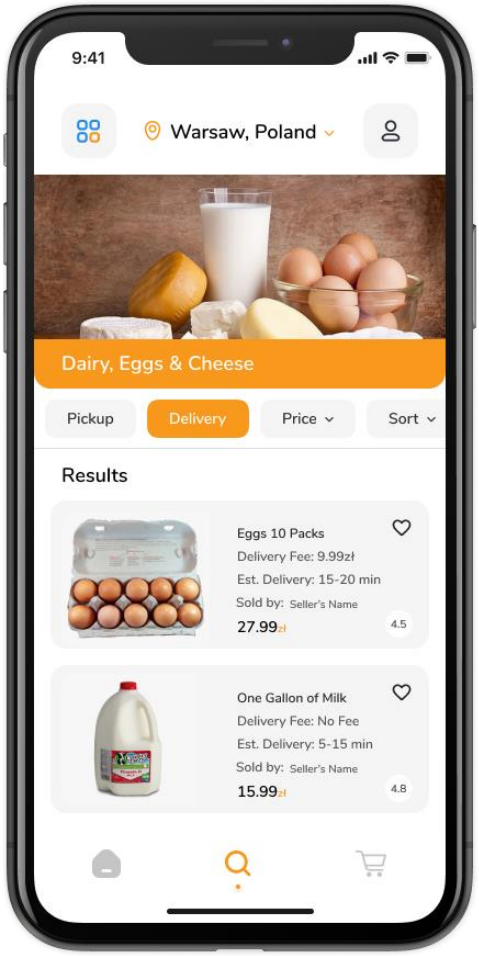
- About section shows application version with social media links to our store.

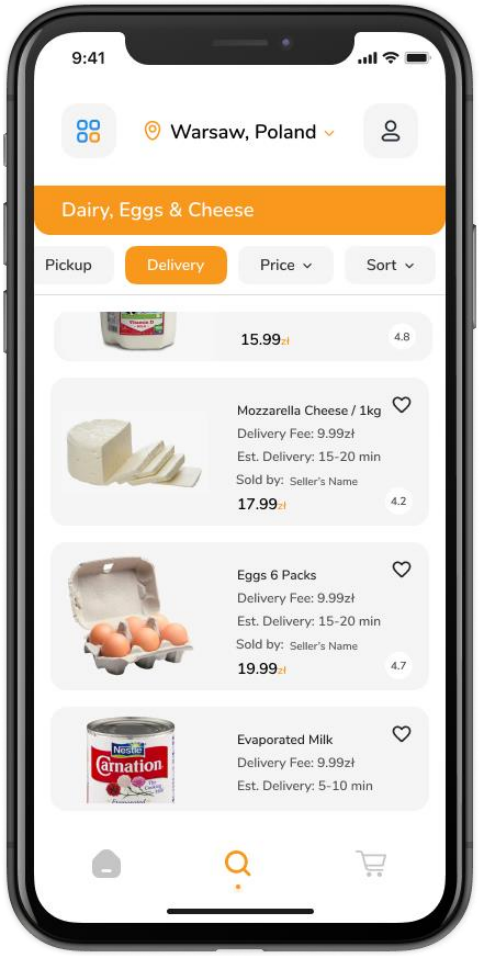


All Categories Page:

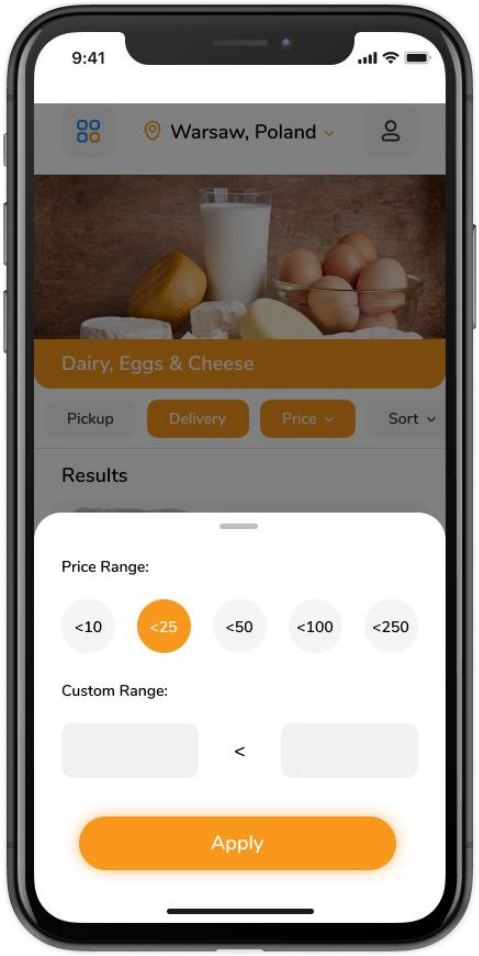
- All categories page can be accessed by opening the top-left icon. Hear users can see the various categories of products covered on the platform.
- It also helps narrow searches by allowing users to open a category they are most interested in quickly.

- Each category is more organized and displays all products that fit into that area, offering filters and a quick way for buyers to find the product they need.
- Users can use filters to narrow their options and locate what they're precisely looking for. These are the following filters available:
    - Products that customers can pick up from markets/warehouses
    - Products that customers can receive by delivery.
    - Filtering by Price.
    - Sorting.

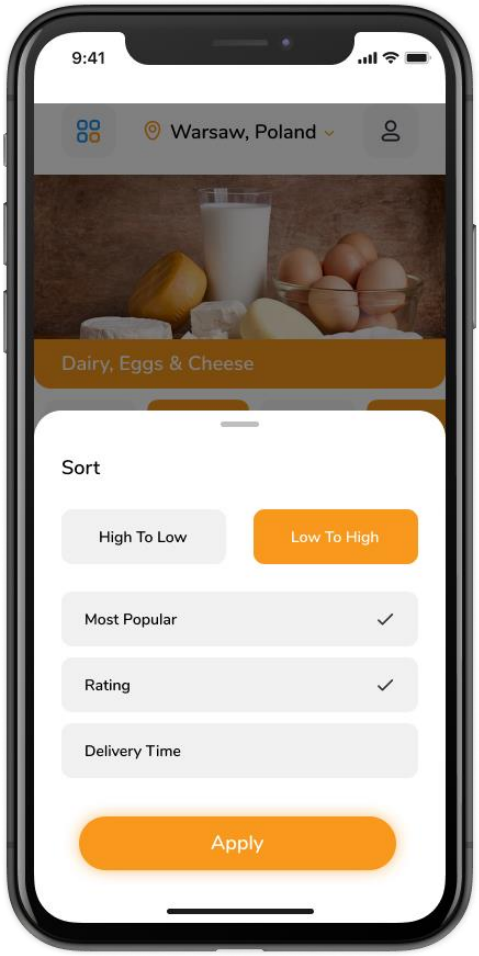

9:41
Warsaw, Poland
Dairy, Eggs & Cheese
Pickup
Delivery
Price
Sort
Results
Eggs 10 Packs
Delivery Fee: 9.99zł
Est. Delivery: 15-20 min
Sold by: Seller's Name
27.99zł
4.5
One Gallon of Milk
Delivery Fee: No Fee
Est. Delivery: 5-15 min
Sold by: Seller's Name
15.99zł
4.8



- Price Filtering can be done with the following options:
    - Products are less than 10 PLN
    - Products are less than 25 PLN
    - Products are less than 50 PLN
    - Products are less than 100 PLN
    - Products are less than 250 PLN
    - Custom Price ranges from the lowest price to the highest price.



- Sorting can be done by the following options:
    - High to Low Price, or Low to High Price
    - Most Popular
    - Rating
    - Delivery Time

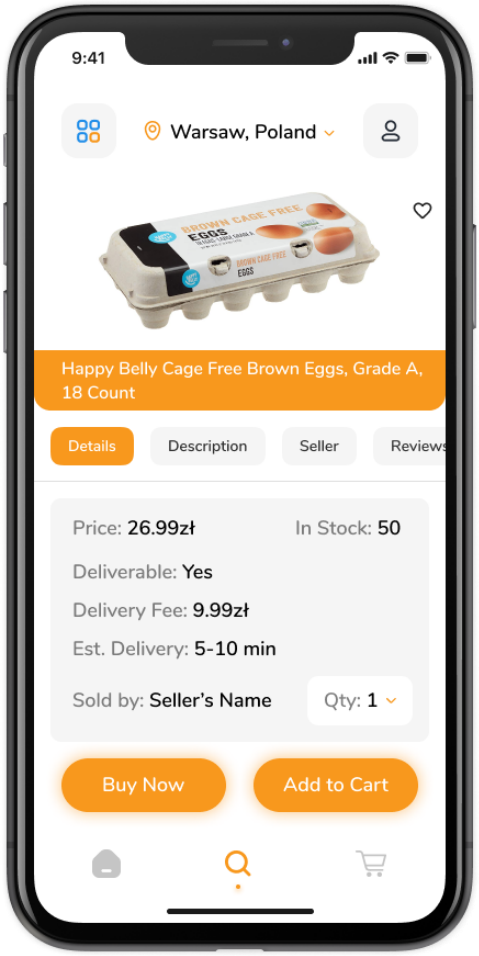


Product View Page:

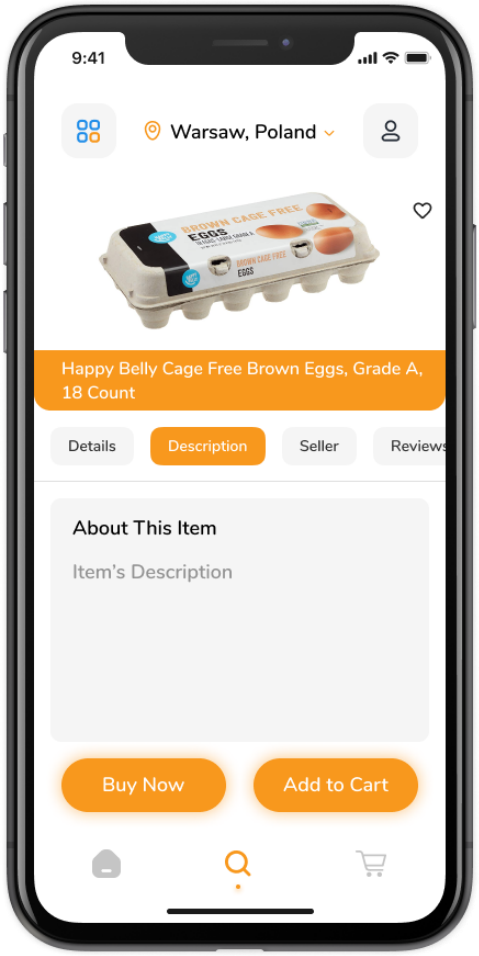
- The product view page includes several tabs that provide additional information about the seller and the product.

Details On Each Tabs Are the Following:

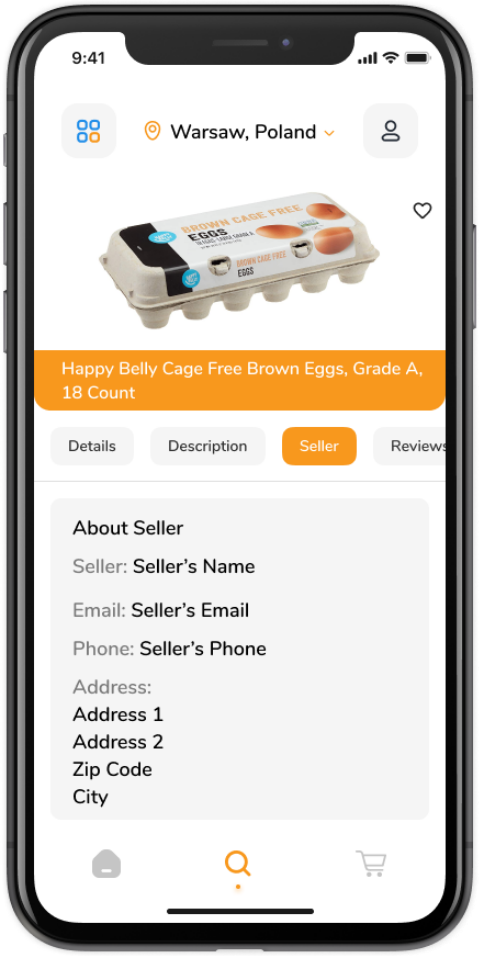
- **Details tab:** it provides information about the product such as:
    - Price
    - Is the product deliverable?
    - Delivery Fee
    - Estimated Delivery Time
    - Is the product in stock?
    - Seller's Name

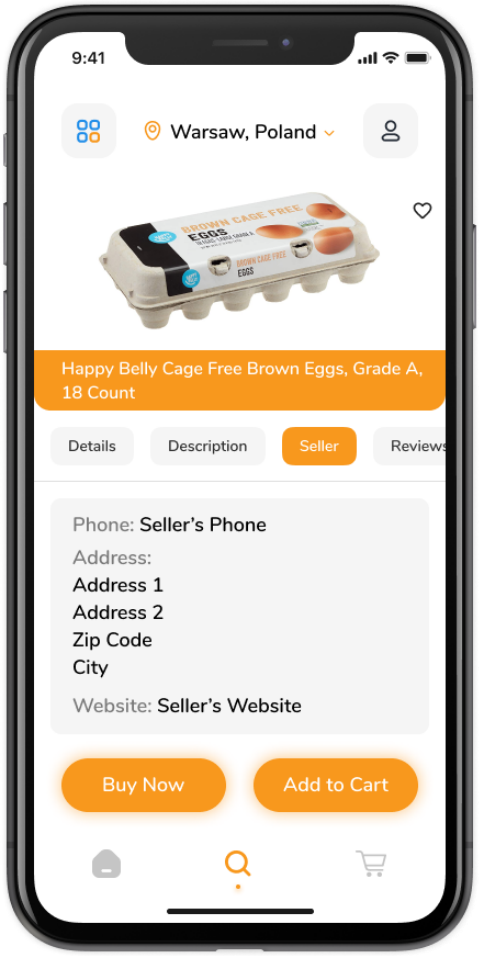


- **Description tab:** It provides more concise details and a brief description of the product.

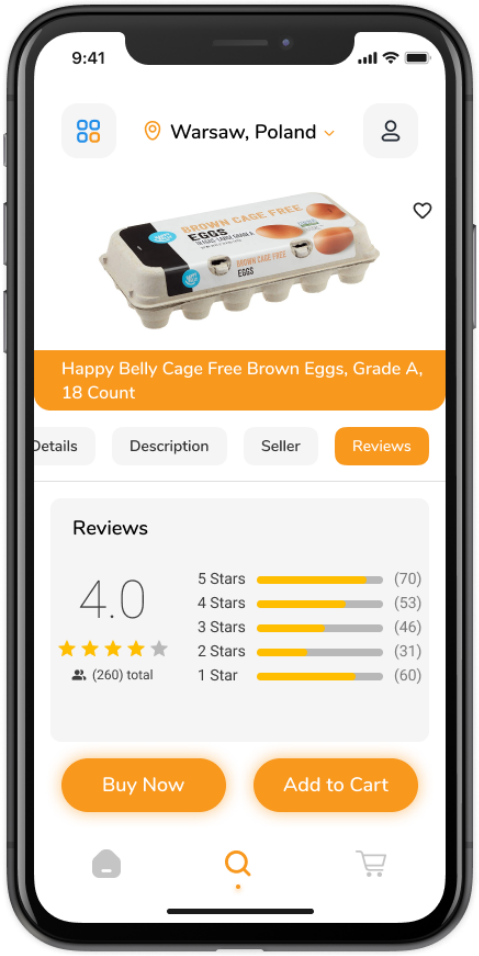


- **Seller tab:** it provides contact information about the seller such as:
    - Name
    - Email Address
    - Phone Number
    - Store address
    - Website


9:41
Warsaw, Poland
BROWN CAGE FREE
EGGS
Happy Belly Cage Free Brown Eggs, Grade A, 18 Count
Details
Description
Seller
Reviews
About Seller
Seller: Seller's Name
Email: Seller's Email
Phone: Seller's Phone
Address:
Address 1
Address 2
Zip Code
City

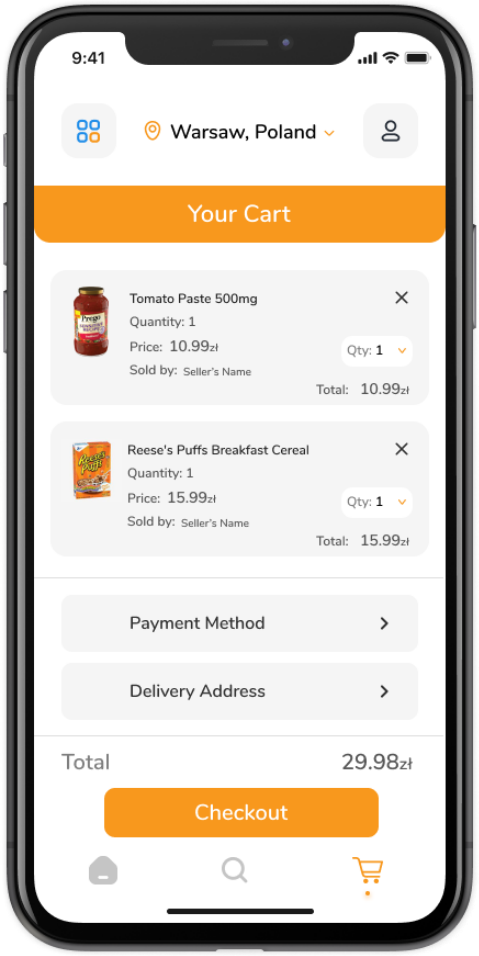


- **Review tab:** It shows the product's rating as well as reviews from other buyers.



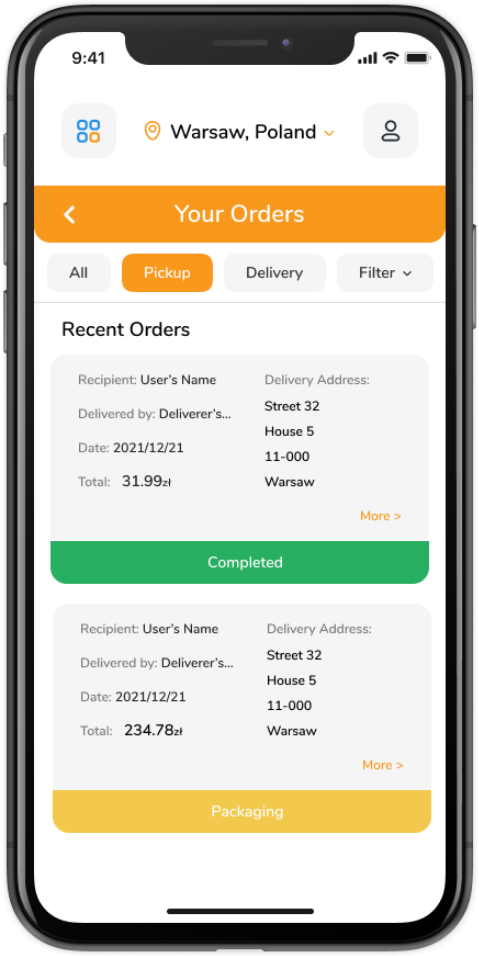
Shopping Cart Page:

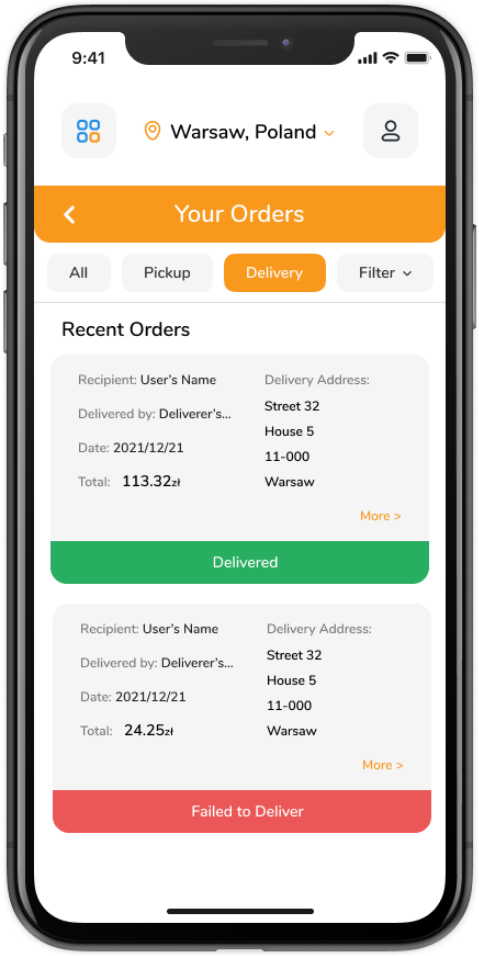
- On the cart page, customers could increase the number of products they have in the cart, or they can entirely remove a product from their cart if they need to.
- The "Payment Method" section of the cart page allows customers to choose their preferred payment method. It also has delivery choices, allowing customers to easily select a delivery address and see the expected delivery time for that destination.



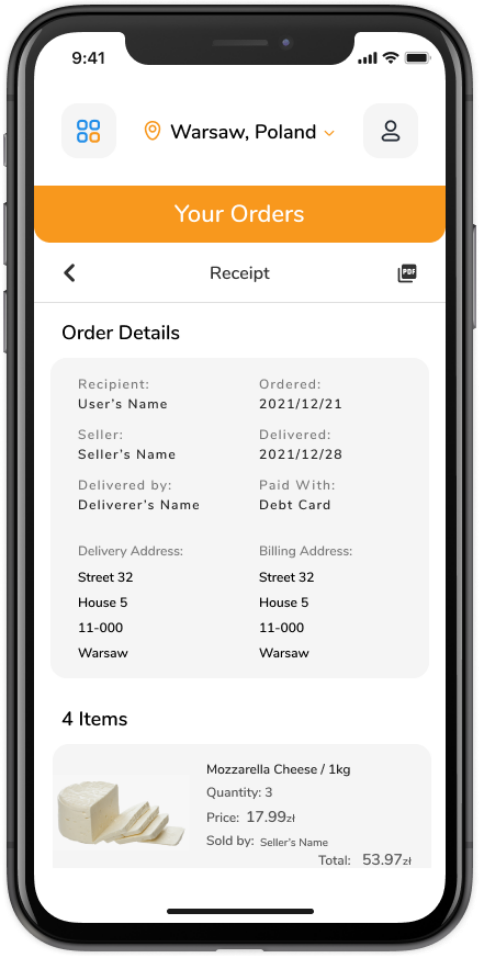
Order History Page (Reference A.2):

- This page gives a lot of information about the order that has been made by the customer.
- The page also provides a convenient way to filter out the user's order history to find the required order.
- The following filters are available:
    - **All**: Displays all the orders regardless of their delivery option.
    - **Pickup**: Displays all the orders that have pickup as a delivery option.
    - **Delivery**: Displays all the orders that have delivery as a delivery option.
    - **Filter**: Displays filter menu to filter out the orders.

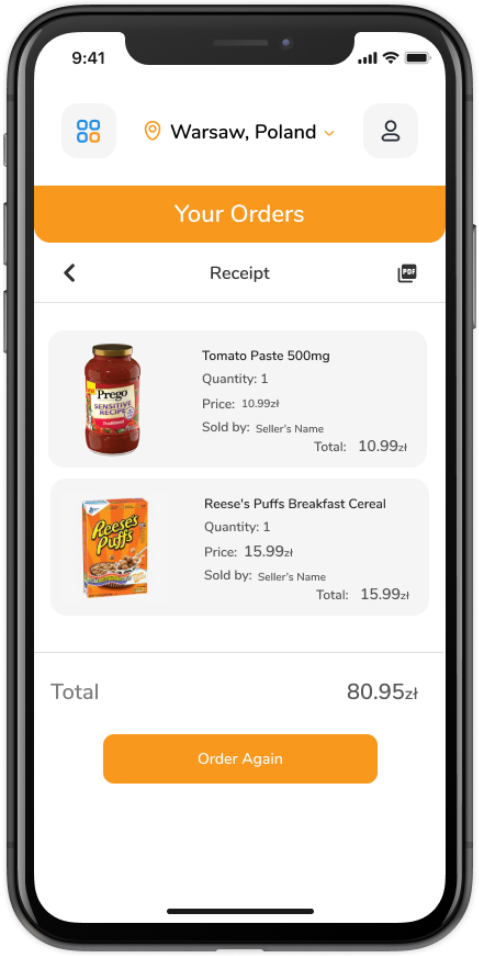

9:41
Warsaw, Poland
Your Orders
All
Pickup
Delivery
Filter
Recent Orders
Recipient: User's Name
Delivered by: Deliverer's...
Date: 2021/12/21
Total: 31.99zł
Delivery Address:
Street 32
House 5
11-000
Warsaw
More >
Completed
Recipient: User's Name
Delivered by: Deliverer's...
Date: 2021/12/21
Total: 234.78zł
Delivery Address:
Street 32
House 5
11-000
Warsaw
More >
Packaging



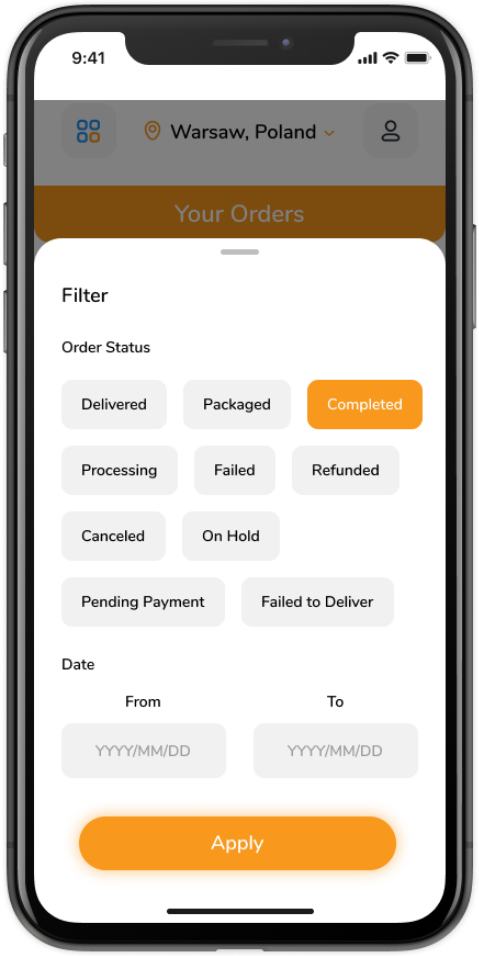
- Customers can also click on an order to display full information about the order, such as:
    - Recipient
    - Seller's Name
    - Delivered by
    - Ordered at
    - Delivered at
    - Paid with
    - Delivery address
    - Billing address



- The application also provides a convenient way for customers to repeat the order without having to worry about placing a new order of the same items.



- Order history offers a filter to easily find the appropriate order that the customer wants to look at. You can filter by Order Status and Date.
- Order Statuses that can be filtered are the following:
    - Delivered
    - Packaged
    - Completed
    - Processing
    - Failed
    - Refunded
    - Canceled
    - On Hold
    - Pending Payment
    - Failed to Deliver


9:41
Warsaw, Poland
Your Orders
Filter
Order Status
Delivered
Packaged
Completed
Processing
Failed
Refunded
Canceled
On Hold
Pending Payment
Failed to Deliver
Date
From
To
YYYY/MM/DD
YYYY/MM/DD
Apply

Security

Introduction

We need some form of communication between the application and the server because the app is exclusively for mobile. To do this, we will use the RESTful API to connect the server and the mobile application.

An API endpoint is a point that receives requests and returns responses. In case of RESTful API, the application sends HTTP requests to API endpoints, and the server returns responses, which the application then processes and displays to users.

EnRoute must be easily scalable to handle heavy traffic and traffic spikes, and it must be secure to protect users data. Public cloud services, container orchestration technologies, and various security scanners help a lot in this regard nowadays. EnRoute uses:

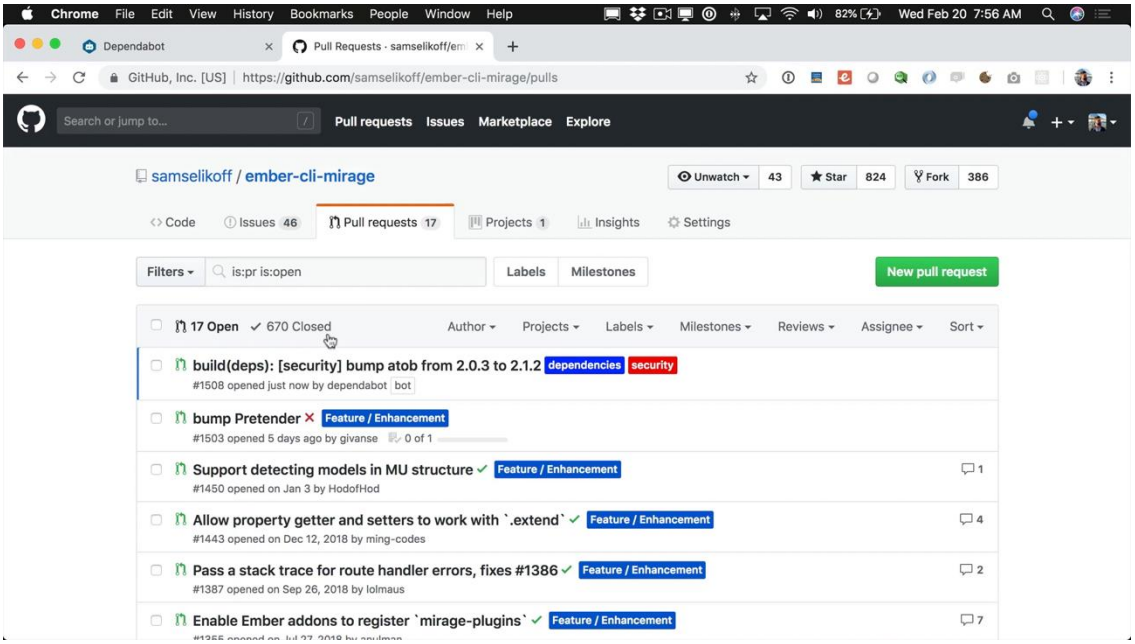
- AWS as a cloud computing service because of it's scalability, available regions, and security possibilities
- Kubernetes as a container orchestration technology to increase application scalability and availability
- Additional security components described below

Security scanning

Dependencies

Dependabot is a GitHub feature, which automatically analyzes project dependencies for known vulnerabilities and automatically creates pull requests with suggested fixes.

In EnRoute it is used to monitor both backend and frontend application dependencies and keep them up to date.



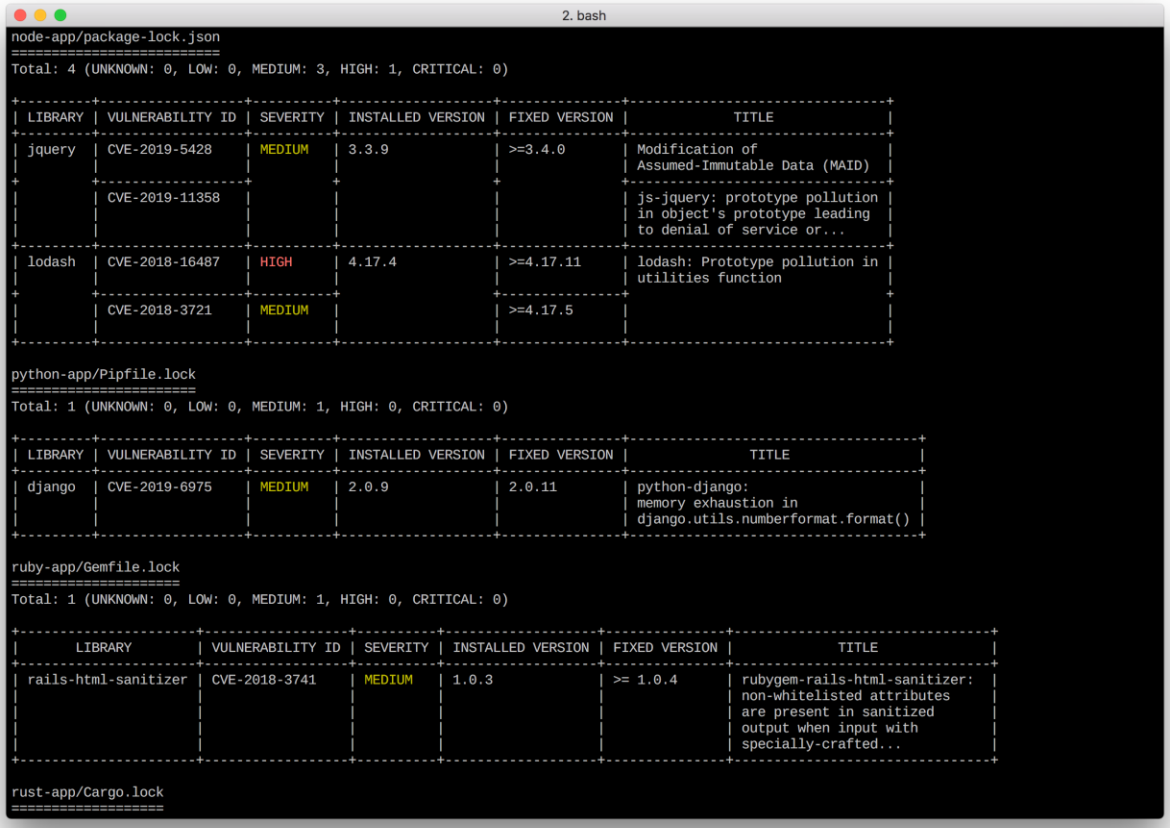
Source: https://embermap.com/video/automate-your-dependency-upgrades-with-dependabot

Docker images

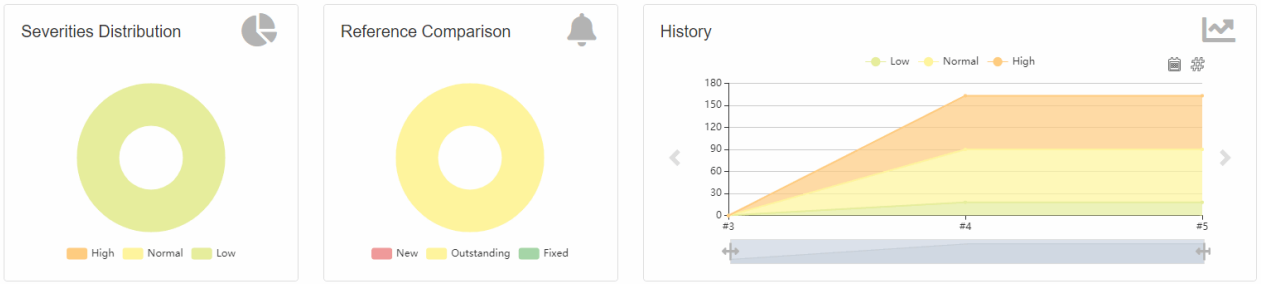
Trivy is a simple, powerful, and open-source CLI tool which allows analyzing container images, file systems, Git repositories, and different configurations. It detects vulnerabilities of OS packages (Alpine, RHEL, CentOS, etc.) and language-specific packages (Bundler, Composer, npm, yarn, etc.). In addition, Trivy scans Infrastructure as Code (IaC) files such as Terraform, Dockerfile and Kubernetes, to detect potential configuration issues that expose your deployments to the risk of attack.

In EnRoute, Trivy is integrated into the CI/CD process based on Jenkins. It scans backend container images for known OS package vulnerabilities (e.g. in various system libraries required to run the backend), Ruby dependencies listed in Gemfile/Gemfile.lock, and frontend dependencies listed in package.json/package-lock.json.

Scanning results are saved into a file, which is then analyzed by the Warnings Next Generation Jenkins plugin. Results are presented as a graph on pipeline pages, and quality gates are configured to fail builds which contain critical known vulnerabilities.



One of the possible Trivy output formats. Source: https://aquasecurity.github.io/trivy/v0.17.0/



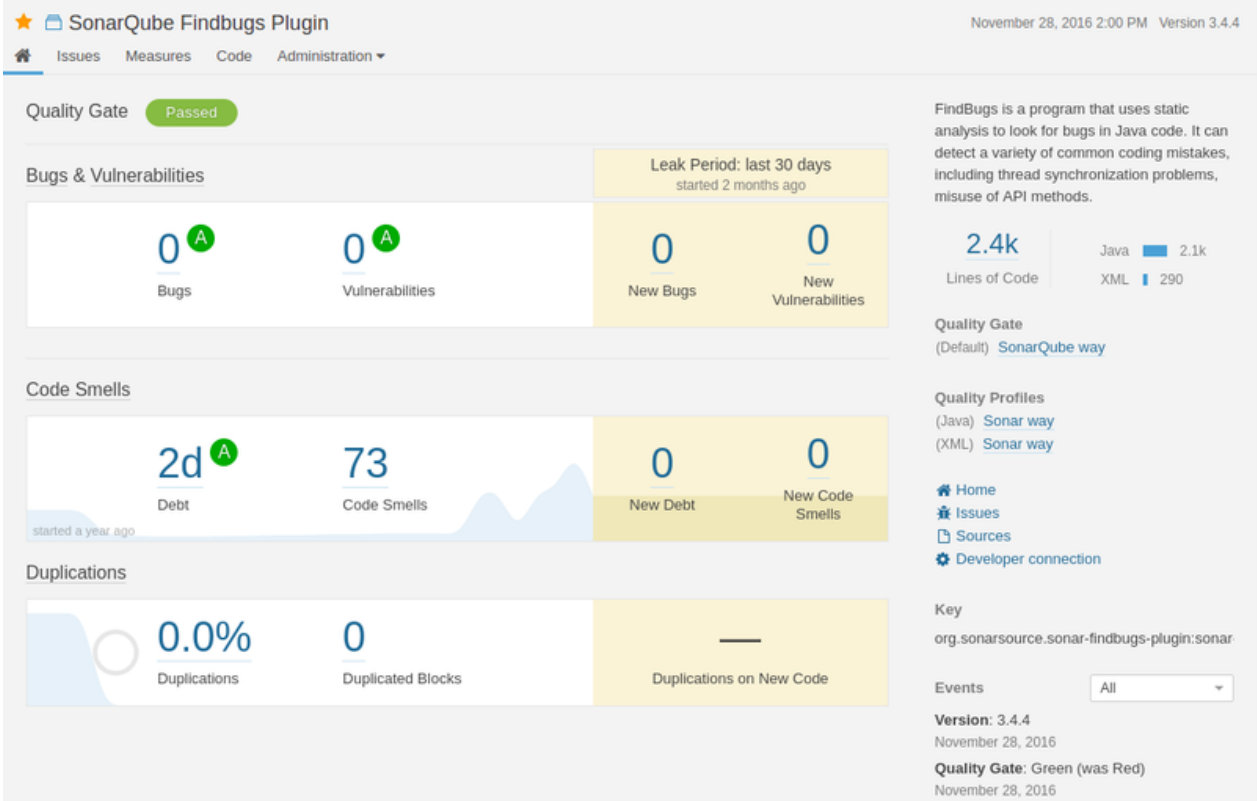
Warnings Next Generation Jenkins Plugin. Source: https://pvs-studio.com/en/blog/posts/0668/

Source code

Source code must be analyzed for quality to maintain low tech debt, and for common errors, which can lead to serious security issues, e.g. not properly validated input which results in SQL Injection vulnerabilities.

SonarQube is a popular and open-source solution, which addresses both issues and supports more than 20 programming languages. It detects bugs, code smells, vulnerabilities, test coverage and shows when an issue was introduced and by whom.

In EnRoute, SonarQube is self-hosted on an AWS EC2 instance and is integrated into the CI/CD pipeline, based on Jenkins. Jenkins runs SonarQube analysis for each deployment, uploads results to a self-hosted SonarQube instance, and fails deployments if test coverage and code quality is below a threshold.



SonarQube Dashboard. Source: https://commons.wikimedia.org/wiki/File:SonarQube_Project_page.png

Peer review

Peer review is a technical control which prevents code entering main Git branches without one or multiple other developers code reviews. It helps in cases such as:

- Developer's workstation is compromised, and an attacker can push vulnerable code to main Git branches, which are automatically deployed, or a change is simply not noticed by anyone.
- A developer makes a mistake and tries to introduce vulnerable code into the repository.

In EnRoute, GitHub policies are configured in such a way that new code must first enter feature branches and then it can be merged into the develop branch. At least 2 approvals from other team members are required to merge a pull request.

Infrastructure

Infrastructure management

Infrastructure as Code (IaC) is a popular practice, which allows managing and provisioning the whole infrastructure via machine-readable definition files. In EnRoute, Terraform is used as a main IaC tool.

Terraform is an industry standard open-source tool, which allows infrastructure to be expressed as code in a simple, human readable language called HCL (HashiCorp Configuration Language). It reads configuration files and provides an execution plan of changes, which can be reviewed for safety and then applied and provisioned. Extensible providers allow Terraform to manage a broad range of resources, including IaaS, PaaS, SaaS, and hardware services.

EnRoute uses Terraform to provision, manage, and update AWS and Kubernetes resources in all regions:

- Encrypted and highly available EKS/RDS/ElastiCache clusters
- Kubernetes access policies and deployments of standard Helm charts
- IAM users
- VPCs
- Route53 domain records
- Encrypted S3 buckets
- Etc.

```
variable "base_network_cidr" {
  default = "10.0.0.0/8"
}

resource "google_compute_network" "example" {
  name                    = "test-network"
  auto_create_subnetworks = false
}

resource "google_compute_subnetwork" "example" {
  count = 4

  name          = "test-subnetwork"
  ip_cidr_range = cidrsubnet(var.base_network_cidr, 4, count.index)
  region        = "us-central1"
  network       = google_compute_network.custom-test.self_link
}
```

Terraform HCL code example. Source: https://www.terraform.io/

Additionally, Terraform is configured to use the S3 backend and lock state using the DynamoDB. Such an approach ensures that each person has an updated infrastructure state and only one person can perform any changes at the same time.

Security scanning

To analyze Terraform code for security issues, EnRoute uses an open-source tool tfsec. It is a static analysis tool, which supports AWS/Azure/GCP/DigitalOcean/OpenStack/CloudStack providers and prevents infrastructure security issues before deploying any changes.

In EnRoute, tfsec is automatically run for all changes in Terraform modules and ensures vulnerable code is not deployed. Found issues must be fixed or a special comment must be added to the code to explicitly ignore issues if there is no possibility or no sense to implement a fix.

tfsec output. Source: https://securityonline.info/tfsec-scan-terraform-code/

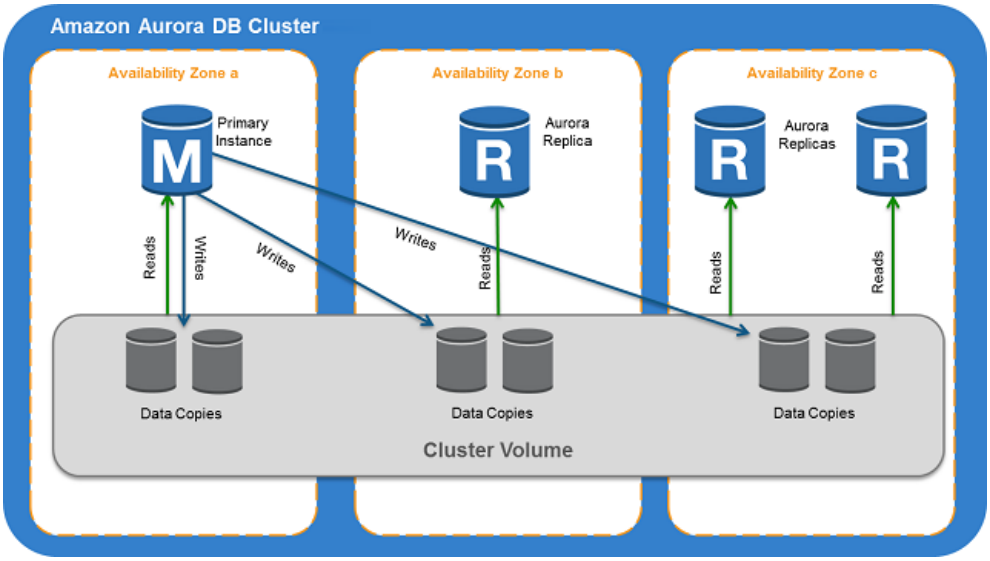
RDBMS

EnRoute uses PostgreSQL as an RDMS. AWS Aurora provides managed MySQL- and PostgreSQL-compatible clusters, which offer increased performance, availability, scalability, point-in-time recovery, patching, and cloning features.

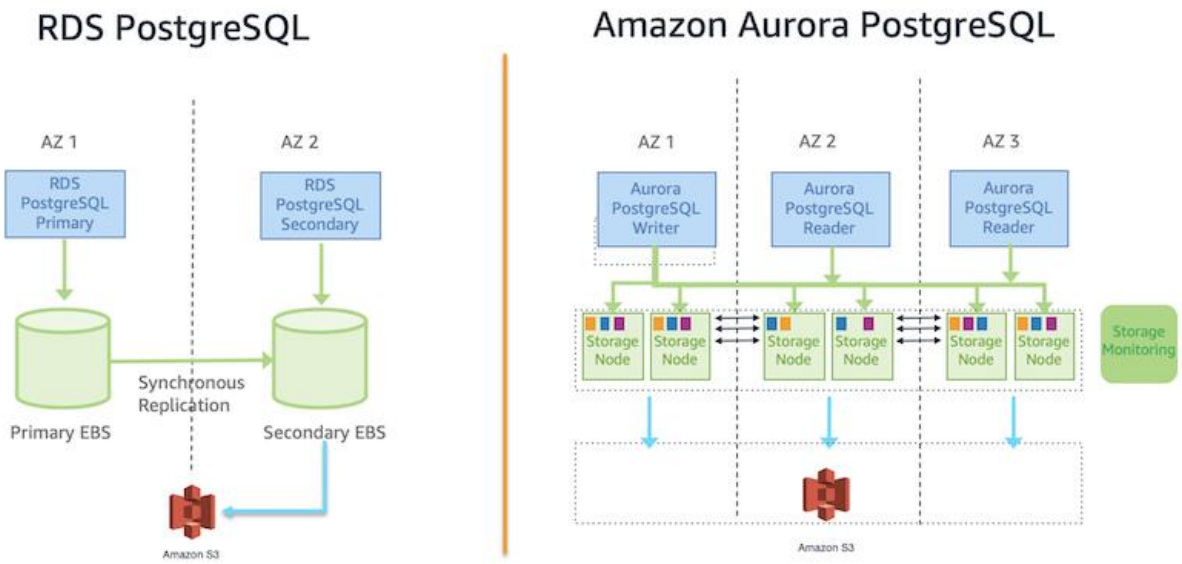
EnRoute provisions privately accessible Aurora clusters, allowing incoming connections from the VPC only. These database clusters are encrypted both at-rest and in-transit.

To increase availability, Aurora clusters are deployed in multiple AWS availability zones. Also, database backups and logs (e.g., slow requests and errors logs) are stored for 1 month.

Additionally, maintenance windows are configured to automate database patching and automatic minor version upgrades.



Aurora architecture overview. Source: https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.html



RDS vs Aurora architecture overview. Source: https://aws.amazon.com/blogs/database/is-amazon-rds-for-postgresql-or-amazon-aurora-postgresql-a-better-choice-for-me/

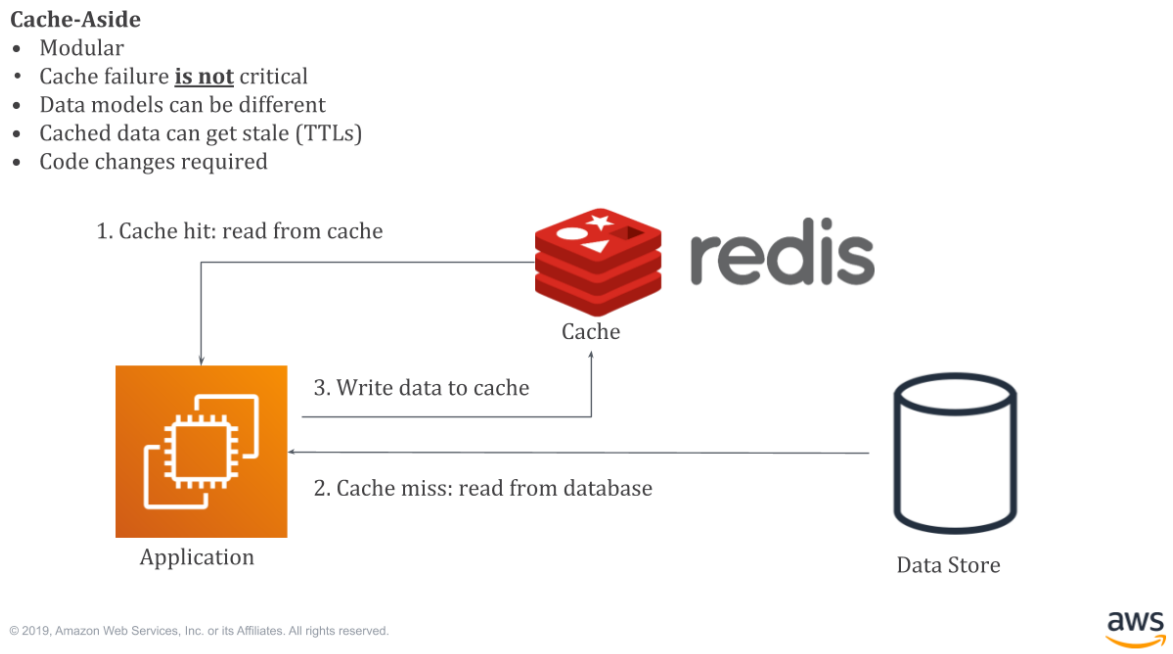
Redis

EnRoute's backend uses Redis as a key-value storage to cache various databases and HTTP requests to decrease response latency and database load. AWS ElastiCache provides managed Redis- and Memcached-compatible clusters with increased availability, scalability, patching, and cloning features.

EnRoute provisions privately accessible ElastiCache clusters, allowing incoming connections from the VPC only. These database clusters are encrypted both at-rest and in-transit.

To increase availability, ElastiCache clusters are deployed in multiple AWS availability zones with automatic failovers configured, in order to switch replica to a master in case of master's failure (e.g., in case of the AZ network issues, node reboot, etc.).

Additionally, maintenance windows are configured to automate database patching.



Backend-Redis communication. Source: https://aws.amazon.com/getting-started/hands-on/boosting-mysql-database-performance-with-amazon-elasticache-for-redis/

OpenSearch

EnRoute's backend uses OpenSearch as a full-text search engine. It provides near real-time search and analytics for all types of data, including structured or unstructured text, numerical data, geospatial data, and so on.

OpenSearch is a community-driven, open-source fork of Elasticsearch and Kibana (derived from Elasticsearch 7.10.2 and Kibana 7.10.2). It also includes features such as enterprise security, alerting, machine learning, SQL, index state management, and more. All the software in the OpenSearch project is released under the Apache License, Version 2.0. The fork was created after Elastic's decision to change the license, and the purpose of that change was mainly to block AWS: Elastic wanted to make it impossible for AWS to create its own competing managed Elasticsearch service.

AWS OpenSearch provides managed clusters with increased availability, scalability, patching, and cloning features. EnRoute provisions privately accessible clusters, allowing incoming

connections from the VPC only. These database clusters are encrypted both at-rest and in-transit. In order to increase availability, OpenSearch clusters consist of 3 nodes (in separate AWS availability zones) with a replica per index. Additionally, maintenance windows are configured to automate database patching.

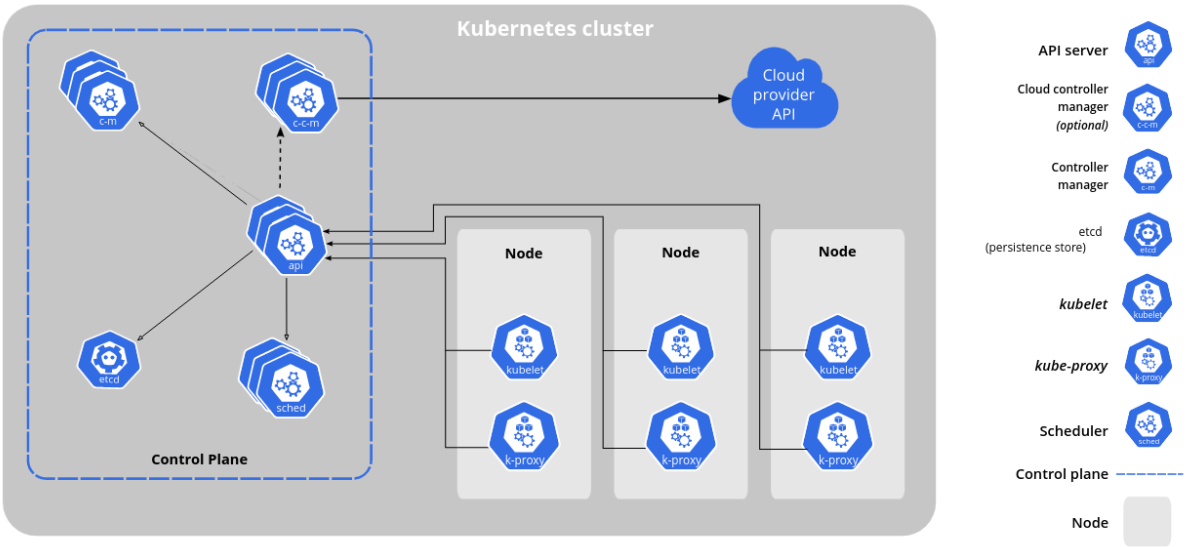
Kubernetes

EnRoute uses Kubernetes (K8s) as a container orchestration platform. Kubernetes is an industry standard open-source system designed to automate application deployments, scaling, and lifecycle management.

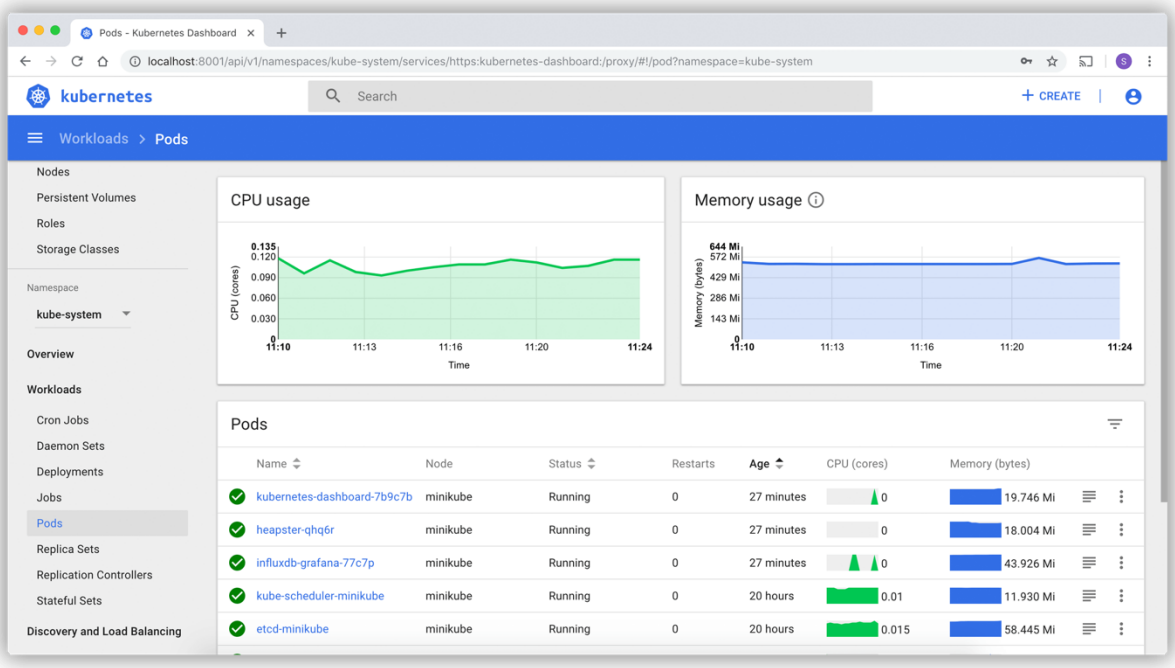
AWS provides a managed Kubernetes service called EKS, which provides a highly available Kubernetes control plane and integrates with other AWS services.

EnRoute configures EKS in such a way that API server accepts incoming connections only from the VPN, or the same VPC. Developers can access EKS clusters using their IAM accounts, and they have limited access to Kubernetes resources based on K8s roles and role bindings.

etcd storage is encrypted by a KMS key. Internal traffic is not encrypted and does not leave a Kubernetes cluster. Also, K8s nodes are deployed to private subnets, and API server logs are stored for 1 month.



Kubernetes architecture. Source: https://kubernetes.io/docs/concepts/overview/components/



Kubernetes dashboard. Source: https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/

S3

Amazon S3 is an object storage service which aims to provide scalability, high availability, and low latency with 99.999999999% durability and between 99.95% to 99.99% availability.

EnRoute uses S3 to store invoices, product photos and videos, customer avatars and other uploads.

EnRoute configures S3 buckets in such a way that objects are encrypted by a KMS key, and all objects by default are private. When objects like invoices have to be downloaded by customers, EnRoute backend generates a presigned URL and returns it to frontend. A presigned URL allows access to a private object only for a specified amount of time and only for specific actions (e.g., only GET to download it).

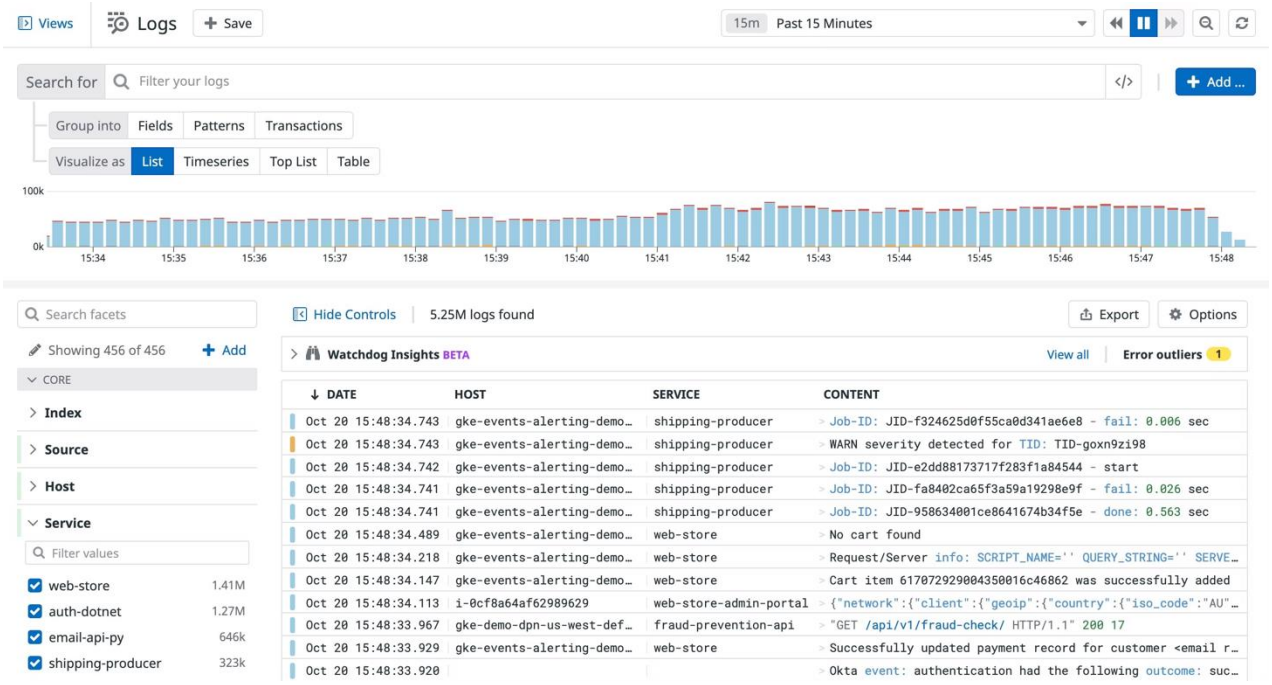
Logging and monitoring

Logging is an essential part of any service, which is used to collect logs from different services e.g. to investigate security incidents, or infrastructure and application issues. Additionally, monitoring is used e.g. to observe performance of various system components and create alerts.

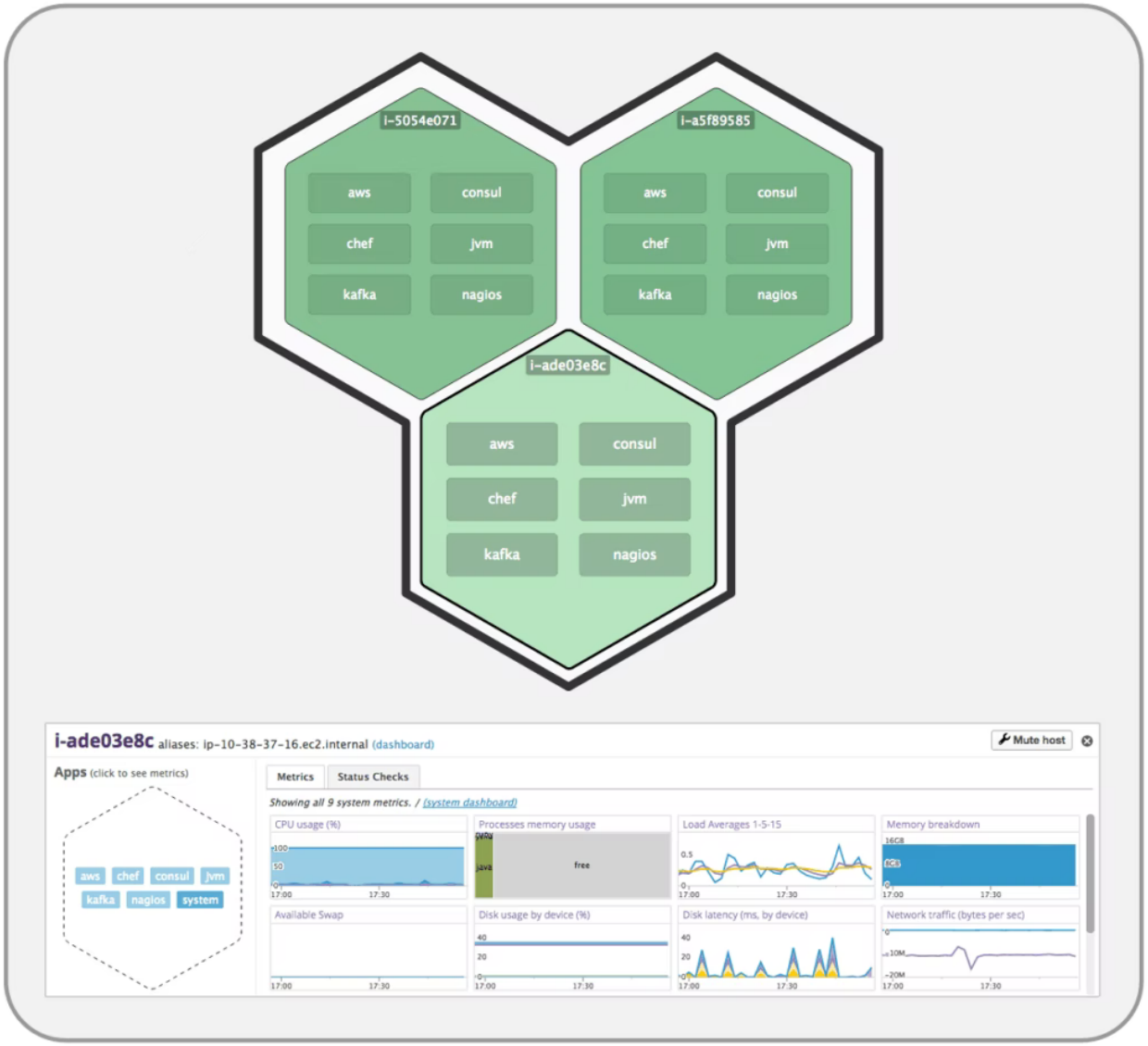
As previously mentioned, EnRoute uses AWS CloudWatch to collect RDS/ElastiCache/EKS logs and metrics, which is mainly used by the operations team. Datadog is used to collect logs and metrics from Kubernetes components, collected data is stored for 1 month, and this tool is used

by both operations and development teams. Datadog is deployed to Kubernetes using the official Helm chart, and it runs an agent on each node collecting their metrics, while also collecting pod logs/metrics.

Additionally, Datadog alerts are configured for critical Kubernetes components like CoreDNS or failing EnRoute backend containers, and Datadog APM is configured to collect additional backend performance insights.



Datadog Log Management. Source: https://docs.datadoghq.com/logs/



Datadog Host Map. Source: https://docs.datadoghq.com/infrastructure/hostmap/

User traffic

User traffic is protected by the HTTPS protocol, which encrypts data in transit and protects against man-in-the-middle attacks. SSL/TLS certificates are issued via AWS Certificate Manager and are managed via Terraform. These certificates use DNS validation method and they are automatically renewed if they are still used in AWS.

Amazon-issued certificates are then attached to Elastic Load Balancers (ELB), which are managed via LoadBalancer Kubernetes objects. Elastic Load Balancers terminate SSL/TLS, and pass traffic to the NGINX Ingress Controller. Additionally, all Kubernetes ingresses automatically perform a redirect to HTTPS, if an ingress was accessed via HTTP.

Backend

Authentication and Authorization

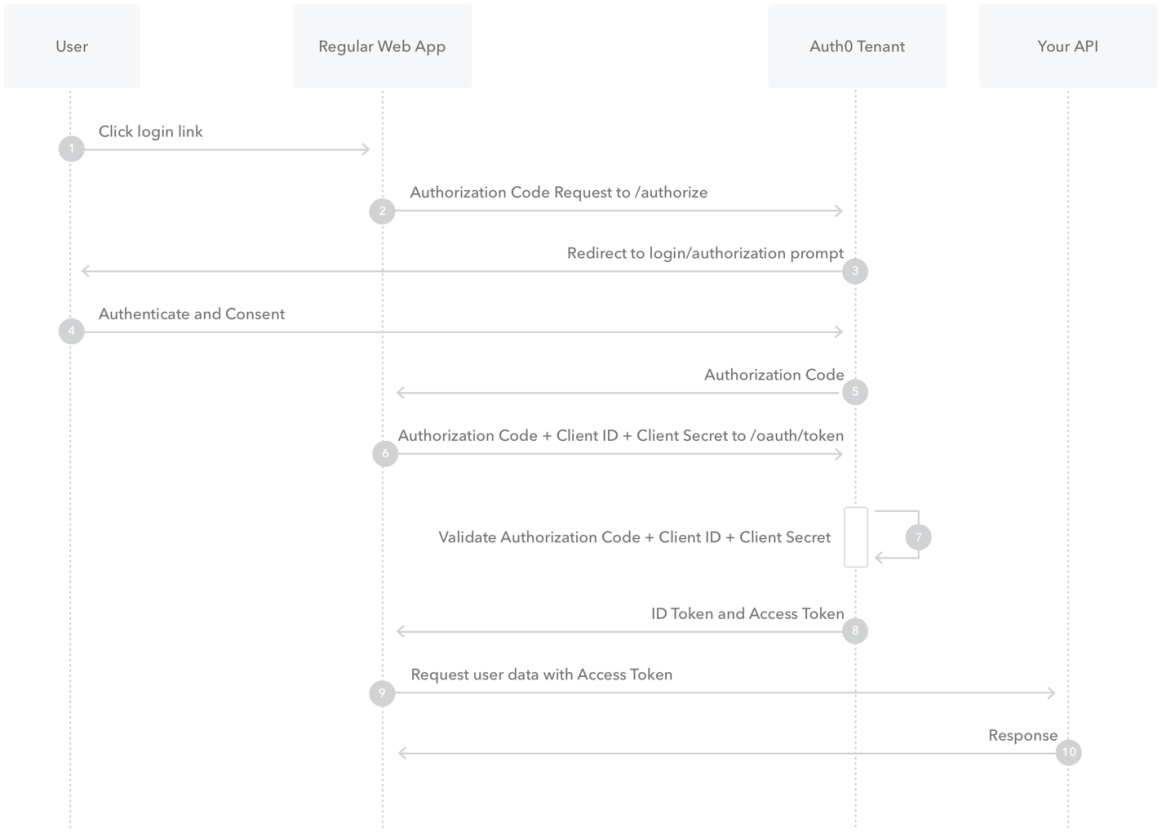
OAuth 2.0 stands for “Open Authorization.” It is a standard designed to allow a website or application to access resources hosted by other web apps on behalf of a user. It replaced OAuth 1.0 in 2012 and is now the de facto industry standard for online authorization. OAuth 2.0 provides consented access and restricts actions of what the client app can perform on resources on behalf of the user, without ever sharing the user's credentials.

Although the web is the main platform for OAuth 2, the specification also describes how to handle this kind of delegated access to other client types (browser-based applications, server-side web applications, native/mobile apps, connected devices, etc.)

OAuth 2.0 is designed only for *authorization*, for granting access to data and features from one application to another. OpenID Connect (OIDC) is a thin layer that sits on top of OAuth 2.0 that adds login and profile information about the person who is logged in. Establishing a login session is often referred to as *authentication*, and information about the person logged in (i.e., the **Resource Owner**) is called *identity*. When an Authorization Server supports OIDC, it is sometimes called an *identity provider*, since it *provides* information about the **Resource Owner** back to the **Client**.

OpenID Connect enables scenarios where one login can be used across multiple applications, also known as *single sign-on* (SSO). For example, an application could support SSO with social networking services such as Facebook or Twitter so that users can choose to leverage a login they already have and are comfortable using.

EnRoute uses Auth0 to manage both authorization and authentication. Auth0 is a flexible, drop-in solution to add authentication and authorization services to applications. This way EnRoute avoids costs, time, and security risks that come with building custom AuthN/AuthZ solutions.



Authorization flow example. Source:
https://auth0.com/docs/authorization/flows/authorization-code-flow

Auth0 is used via official SDKs both for clients using mobile apps, and for administrators using JS frontend – those are 2 different Auth0 tenants (logically isolated groups of users who share common access requirements with specific privileges). User data is stored by Auth0, and users can login both using email/password and social networks.

HTTP verbs

Only POST, PUT, PATCH or DELETE HTTP verbs are used to make any changes, as GET requests, for example, are usually not covered by anti-CSRF mechanisms.

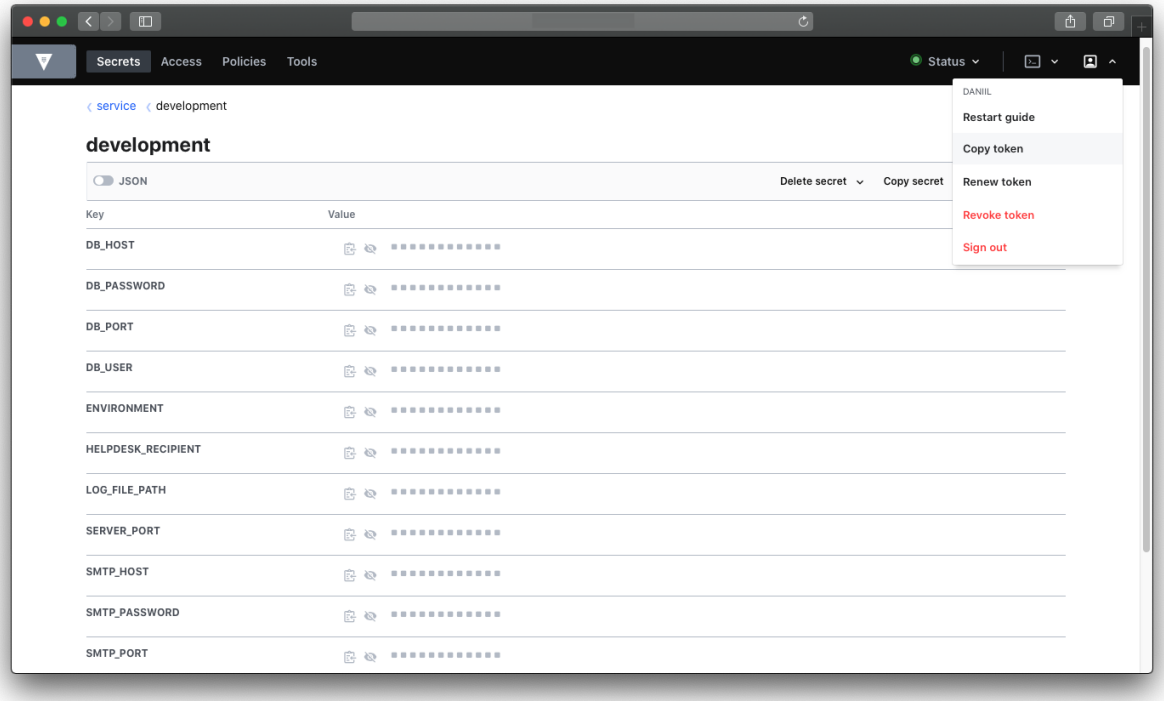
Configuration

Backend performs configuration via environment variables. Such an approach allows to build a single artifact (container image), and deploy it to multiple environments (e.g., dev, staging and production). Also, it lowers the risk of exposing API credentials, for example by accidentally pushing them to the Git repository and allows to configure various access policies to secrets of different environments.

EnRoute stores environment variables in a HashiCorp Vault instance. It is an industry standard open-source system designed to store secrets and access them in secure ways.

In EnRoute, environment variables are pulled by Jenkins during deployments only, and then they are deployed as Kubernetes secrets along with other Kubernetes manifests to deploy the backend. Jenkins uses the AppRole Vault authentication method, which generates a single-use token to download secrets before a deployment.

Additionally, developers are granted limited access to the Vault and last 10 secret versions are stored. Vault is deployed to Kubernetes using the official Helm chart, it uses S3 as a storage backend, and performs an auto-unseal operation using a KMS key.



HashiCorp Vault KV GUI. Source: https://mrpopov.com/posts/hashicorp-vault-as-environment-manager/

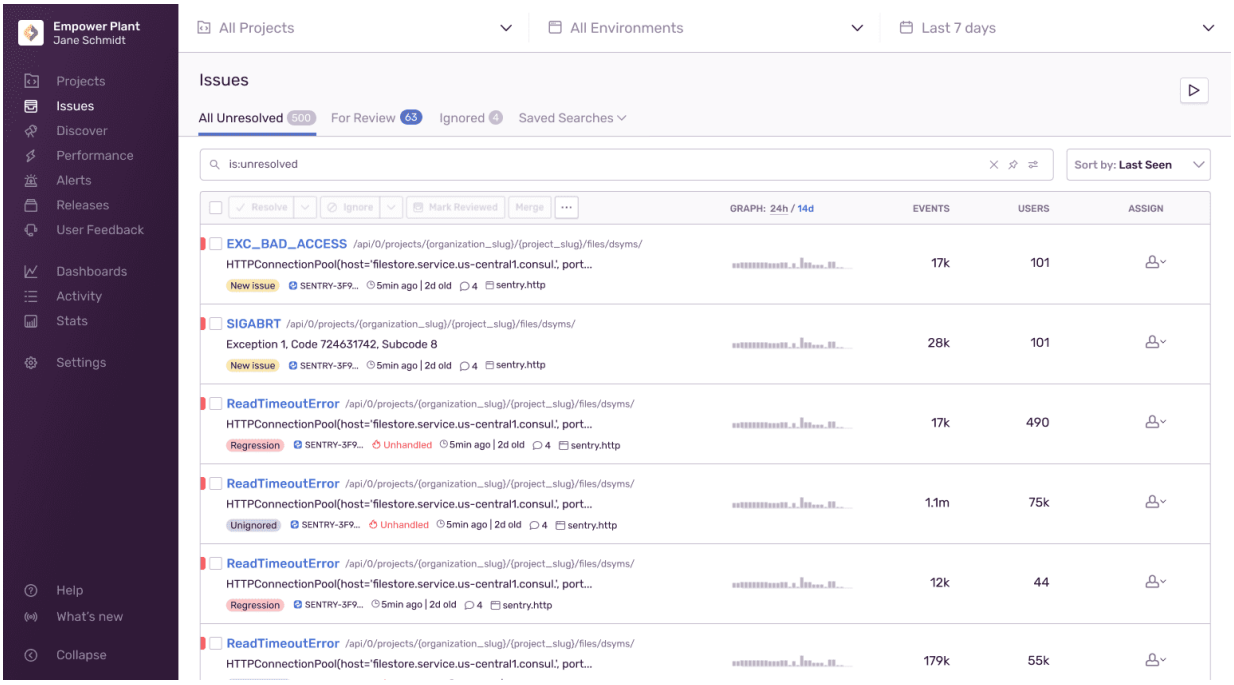
Exceptions catching

Error tracking is important for each stage of product development and maintenance. Anything can go wrong from device compatibility, network issues to unknown ways users are using the app and unknown bugs in an application. To track which error was thrown in the live app both on frontend and backend, it is best to mitigate issues in early stages.

Sentry is one of the most popular solutions which addresses problems like real-time error tracking, deployments tracking, performance monitoring, and so on. Sentry natively integrates with various frontend and backend platforms like .NET, Android, Spring/Spring Boot, React, Angular, Kotlin, Node, Ruby, PHP, and many more. EnRoute uses the following features of Sentry in all applications:

1. Notifying operations and development teams via email and Slack about new errors or old ones resurface.
2. Providing operations and development teams with specific info like stack traces, current environment configuration, and input data.
3. Providing operations and development teams with additional information like current URL, device parameters (OS, User Agent, hardware), where errors happen, how often they happen, and who is affected.
4. Assigning issues to specific team members to observe and fix the issues.

This way no exceptions and errors are lost, all issues are monitored in real-time, and they can be correlated with specific releases or even git commits.



Sentry issues view. Source: https://docs.sentry.io/product/issues/

Database access

One of the challenges of using object-oriented programming (OOP) languages and databases is the complexity of aligning the programming code with database structures. An object-relational mapper provides an object-oriented layer between relational databases and object-oriented programming languages without having to write SQL queries. It standardizes interfaces reducing boilerplate and speeding development time.

With ORM:

1. Code is far easier to update and maintain.
2. Alot of things are done out of the box, e.g., handling underlying database connections and localization.

3. SQL injections are a lot more difficult, because queries are prepared and sanitized.
4. When you change the underlying database (e.g., from MySQL to PostgreSQL), you (almost) do not have to rewrite code.

EnRoute uses Ruby on Rails, which offers Active Record ORM. Also, Active Record is used to manage database migrations.

Object storage

S3 is accessed via official AWS SDK and unique IAM users. Each environment (dev, staging, production) has a designated IAM user which has access only to specific S3 buckets and related KMS encryption keys.

Inputs

EnRoute validates all inputs as strictly as possible. Proper validation makes many vulnerabilities challenging to find and exploit. Invalid input is rejected, and not sanitized.

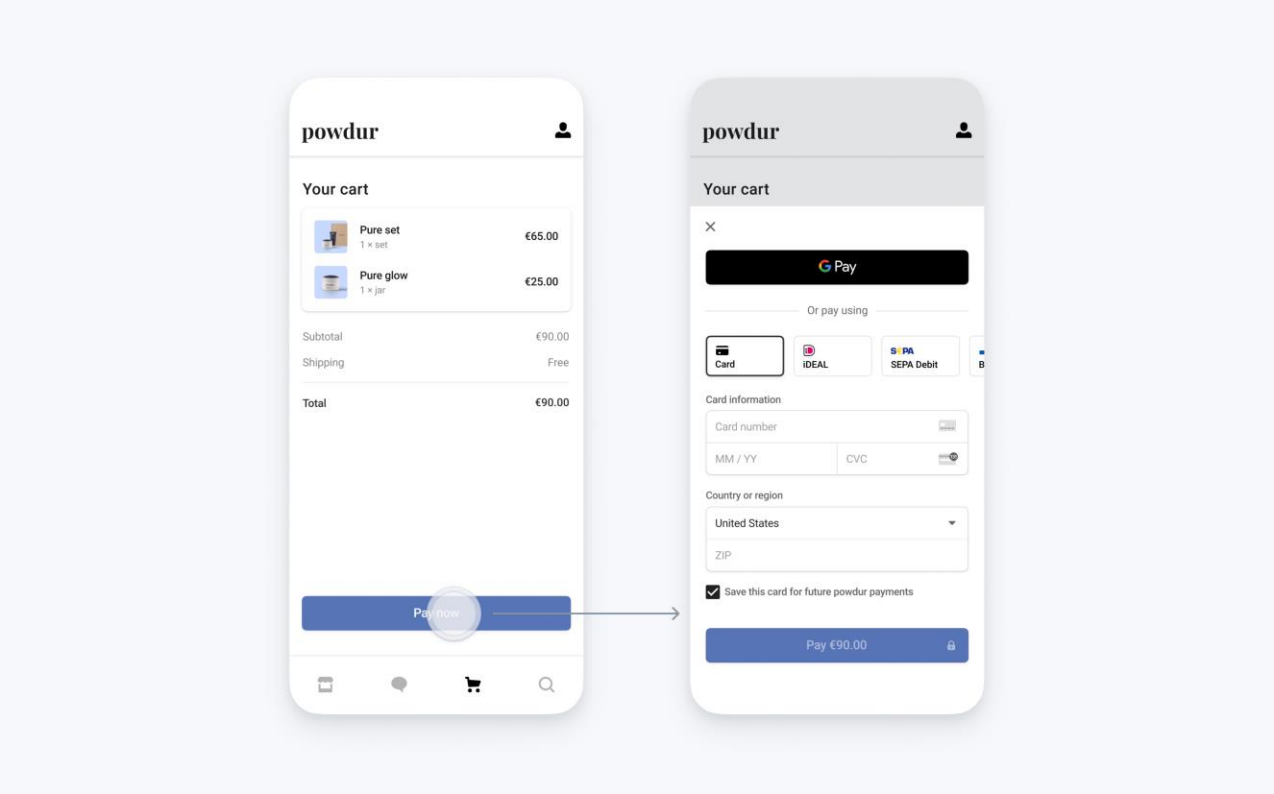
1. Restrictive data types are used as often as possible. DateTime for dates, Integer for numbers, Enums for lists of possible values, and so on.
2. When String must be used, a length limit is set if possible.
3. When String must be used, character set is restricted to the minimum.
4. When processing JSONs, they are validated against a strict JSON schema before proceeding with deserialization.
5. When processing XMLs, they are validated against a strict XML schema before proceeding with deserialization.

Payments

Implementing and maintaining your own payment solution is a very expensive and time-consuming task, and the number of vulnerabilities merchants might face is constantly growing. Instead, popular payment gateways solutions should be preferred.

A payment gateway as a merchant service that processes credit card payments for ecommerce sites and traditional brick and mortar stores. Popular payment gateways include PayPal/Braintree, Stripe, and Square. Such solutions encrypt data for exclusive use between seller and buyer, send an authorization request to a financial institution, and allow a website and interface to proceed to a next action when an authorization request is fulfilled.

EnRoute uses Stripe as a payment gateway. Stripe offers payment processing software and application programming interfaces (APIs) for e-commerce websites and mobile applications. Also, Stripe integrates well with mobile platforms using official SDKs.



Stripe checkout example. Source: https://stripe.com/docs/payments/accept-a-payment

Notifications

Push notifications are short messages that pop up on the user's mobile or desktop, nudging them to take some action. EnRoute uses AWS SNS to deliver notifications to users' Android and iOS devices.

AWS SNS is configured to deliver Android notifications using Firebase Cloud Messaging, and iOS notifications using Apple Push Notification Service. Also, AWS SNS topics are encrypted with a KMS key.

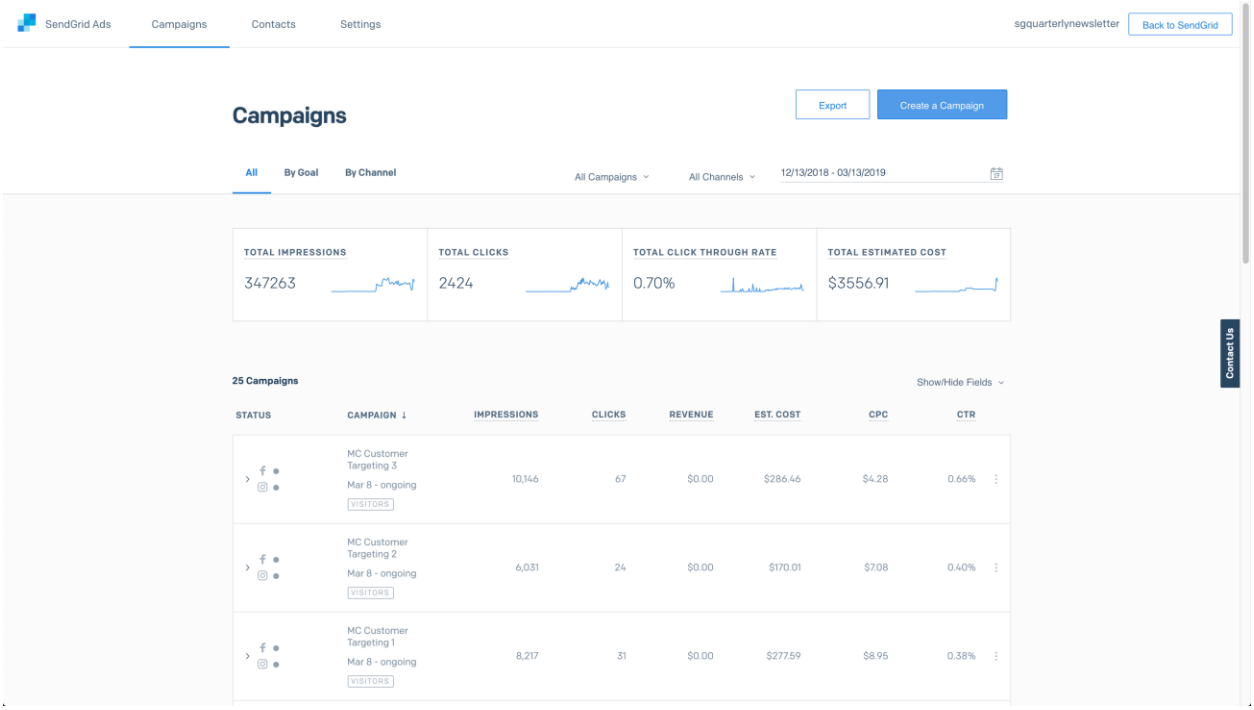
Mailing

Using a managed mailing service offers the following benefits:

1. Avoiding relegation to the spam folder and rejection by mail servers by reputation monitoring.
2. Handling large email volumes.
3. Tracking email performance in real-time and verifying whether emails have reached their destination.
4. Avoiding server maintenance and security headaches.
5. Easy templates management.

EnRoute uses Twilio SendGrid to send all emails, including marketing campaigns. Communication with SendGrid is encrypted and SendGrid is accessed via Web API instead of SMTP because of the following benefits:

1. ISPs might block outbound mail ports by default, e.g., SMTP ports 25/587/465.
2. Web API requires less messages for communication and is significantly faster than SMTP in case of large round-trip time (RTT).



SendGrid Campaigns example. Source: https://www.getapp.com/it-communications-software/a/sendgrid-cloud-based-email-infrastructure/

Pentesting

Automated tests manage to catch most security issues prior to production deployments, but there may still be unnoticed issues. Experienced pentesters should be employed to minimize the risks. These types of ethical hackers attempt to break into the application to detect vulnerabilities and find potential attack vectors with the aim of protecting the system from a real attack.

EnRoute employs external pentesters on a yearly basis to verify the application's security.