Small Business P ortal

Runbook

Document History

|  |  |  |  |
| --- | --- | --- | --- |
| Ver. | Date | Change Description | Name |
| 1.0 | 06/09/2023 | Initial Version | Patrick Killian |
| 1.1 | 06/23/2023 | Filling in gaps/links that were missing in initial rough draft | Patrick Killian |
|  |  |  |  |

Document Review

|  |  |  |
| --- | --- | --- |
| Name/Title | Signature | Date |
|  |  |  |

Document Location

|  |  |
| --- | --- |
| Run Book Name | Link to Artifacts Library |
|  |  |

Contacts

|  |  |  |
| --- | --- | --- |
| Contact Name | Role | Phone Number |
|  | Program Manager |  |
|  | Architect |  |
|  | Portfolio Manager |  |

Document Issues and Notes

Table of Contents

Introduction 4

Acronyms 4

Distribution 5

Source Code 5

Overview 6

Interfacing Systems 9

Informed Delivery (RMIN) 9

Azure 9

Software Architecture 10

Hardware Architecture 13

Load Balancer 13

Procedures 14

Cluster Configuration Procedure 14

Application Server Setup Procedure 15

DNS Settings 15

Web Server Customization Procedure 16

Update Procedure 16

Troubleshooting and Reporting 17

# Introduction

This document provides a technical overview of the Small Business Portal project. It covers the physical architecture and software design. The bulk of the document provides configuration information and installation instructions. This document does not cover business requirements. The intended audience for this document is developers, project managers, or stakeholders.

### Acronyms

The following abbreviations are used throughout this document. Other abbreviations are defined at first use.

AMS – Address Matching System

USPS – United States Postal Service

ZIP11 – An 11-digit code identifying most of the mailboxes in the United States. The ZIP11 combines the ZIP+4 code with a 2-digit code for a specific mailbox.

IV – Informed Visibility

ID – Informed Delivery

IDE – Integrated Development Environment

IDM – Informed Delivery Mobile

ORM – Object relational mapping

DTO – Data Transfer Object

## Distribution

### Source Code

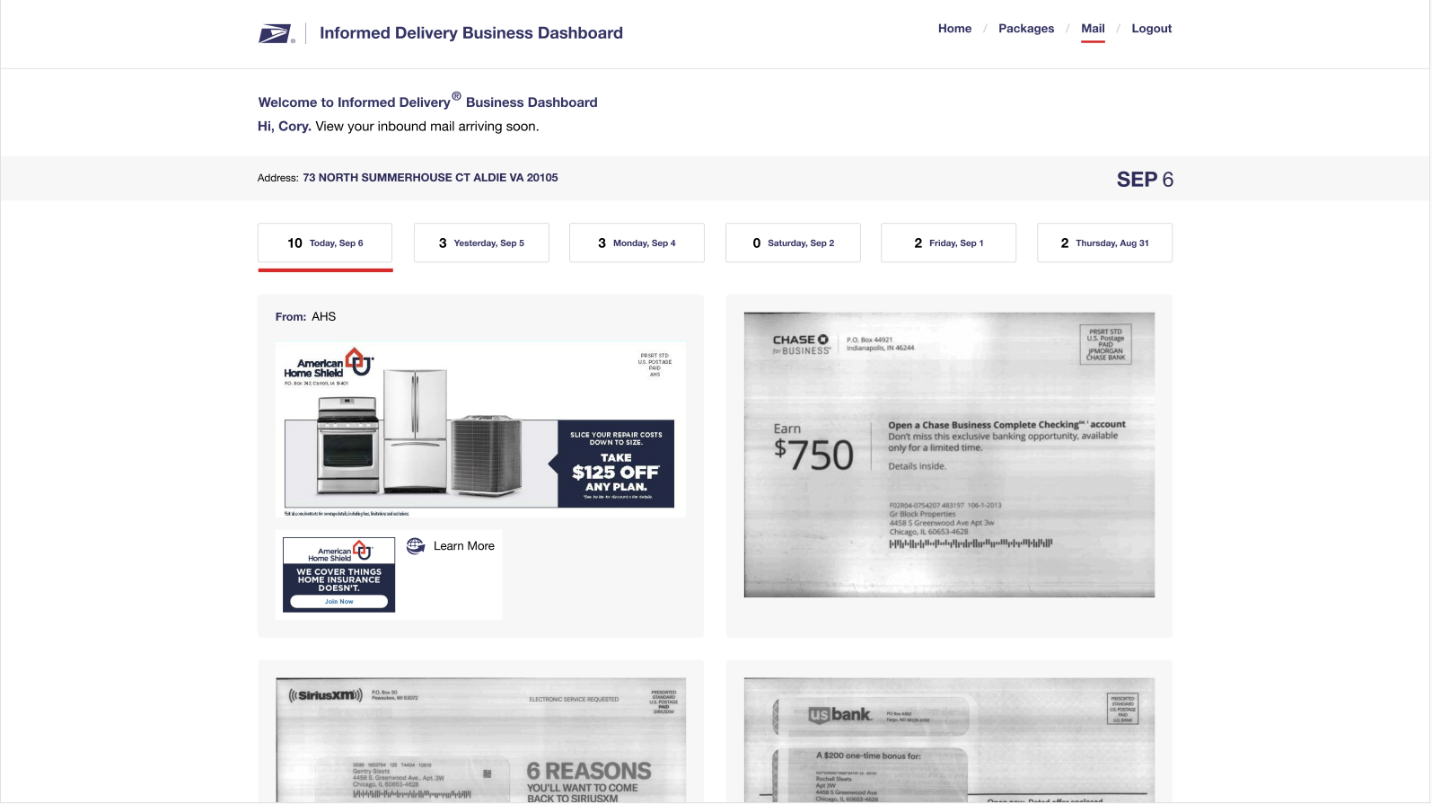
The project source code is currently located in Git repositories in Azure DevOps:

|  |  |  |
| --- | --- | --- |
| **Repository** | **Description** | **URL** |
| informed-delivery-services | Backend web services | git@ssh.dev.azure.com:v3/postaldirect/ID%20Mobile%20Application/informed-delivery-services |
| informed-delivery-services-push-notifications | Backend push service – currently unused by this application but used by IDM project | git@ssh.dev.azure.com:v3/postaldirect/ID%20Mobile%20Application/informed-delivery-services-push-notifications |
| Informedinnovations\_web | Front end dashboard | https://dev.azure.com/postaldirect/informed\_innovations/\_git/informedinnovations\_web |

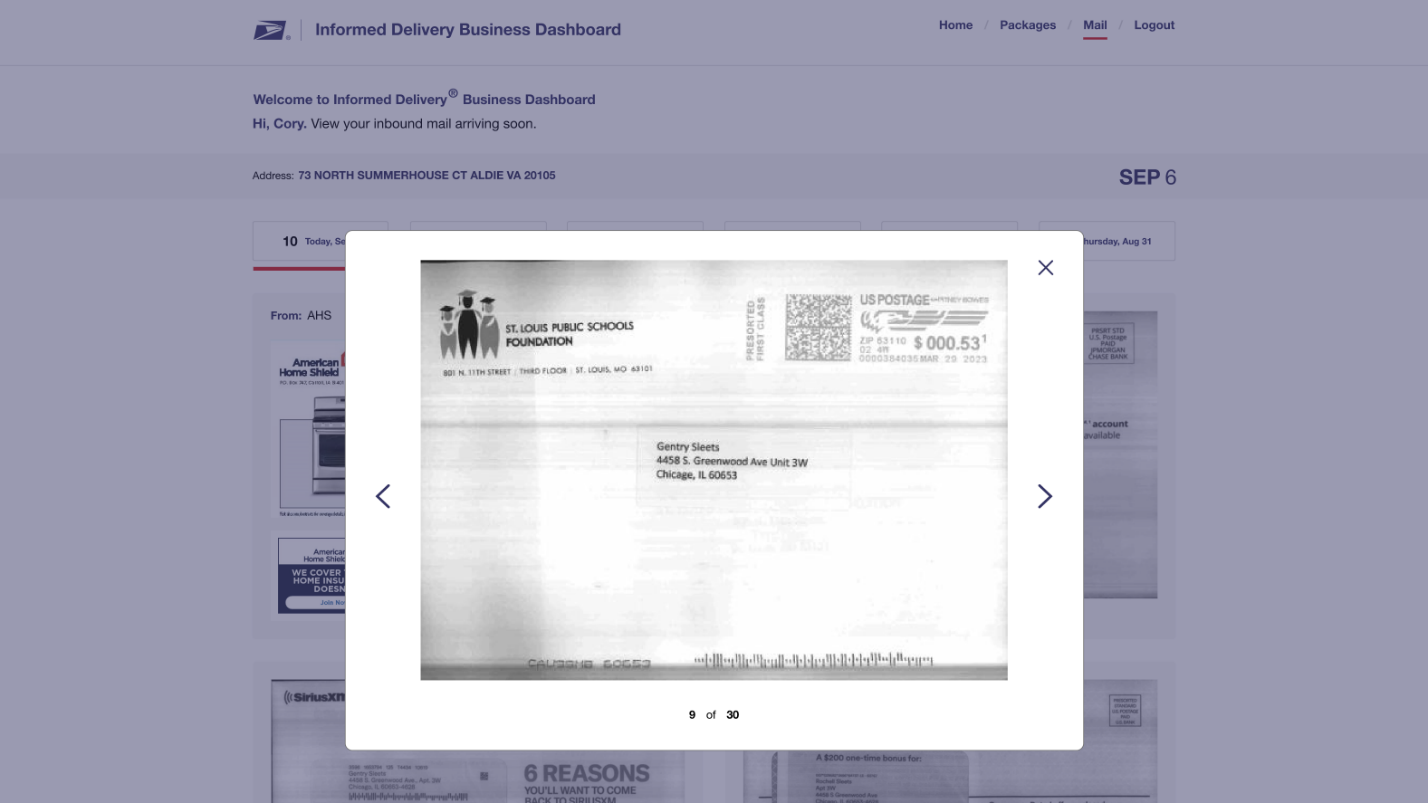
# Overview

Small Business Portal is a web application that allows small business users to access a business focused dashboard containing relevant information from Informed Delivery.

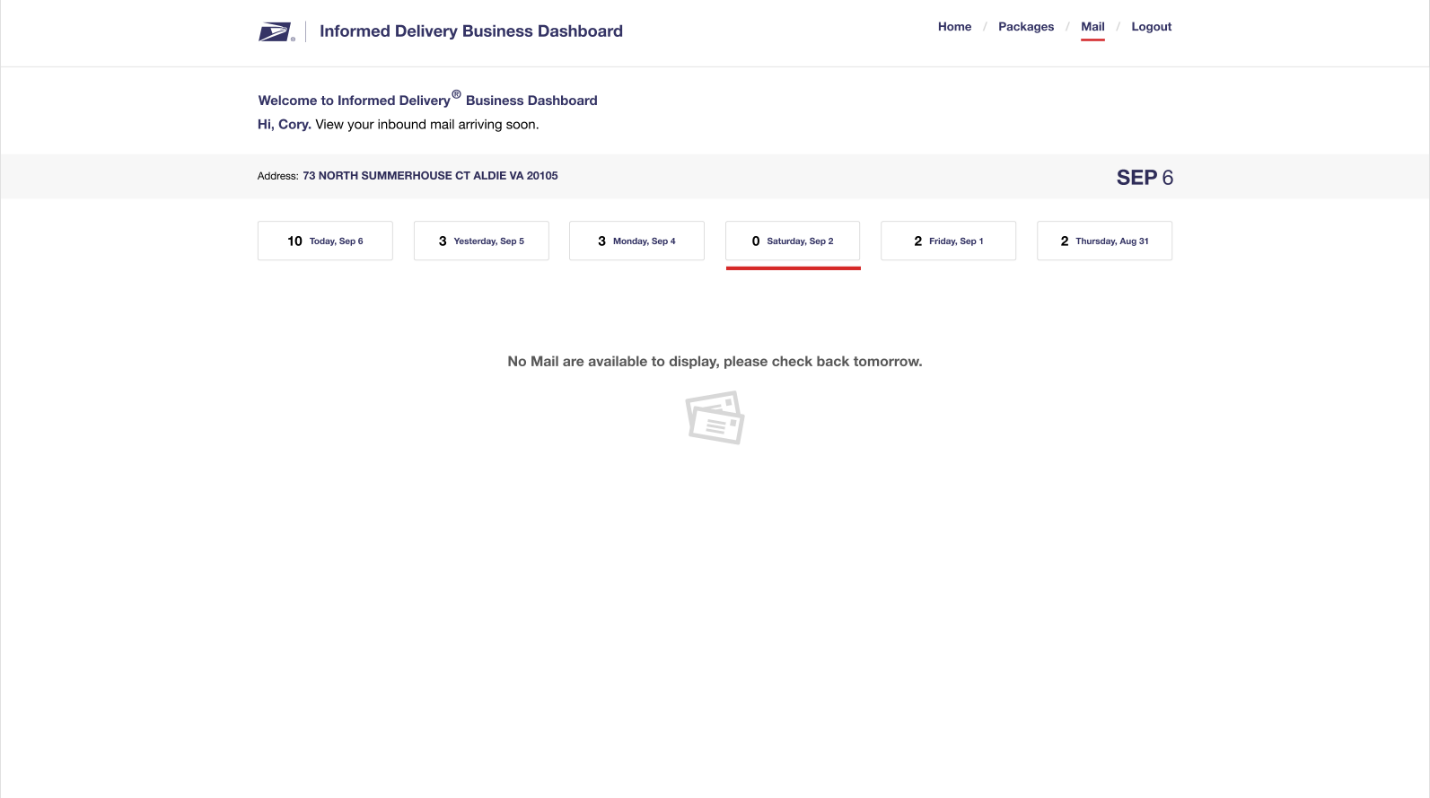
Example of dashboard with mailpieces:



Example of image carousel modal window:



Example of dashboard with no mail for the day:



The web application was developed using {front end framework info}. The website is entirely driven by back end services. The services are implemented in Java 17 as REST endpoints using Java WebFlux.

The services communicate with a temporary backend that scrapes forwarded Informed Delivery daily digest emails for its data. This will be replaced with an API provided by the USPS Informed Delivery team once they have completed development of it.

The platform is containerized and cloud agnostic. Development environment is currently running in an Azure Kubernetes Service cluster.

## Interfacing Systems

### Informed Delivery (RMIN)

Informed Delivery provides mailpiece and ride-along images being delivered to a user’s mailbox.

### Azure

The application integrates with several Azure services, including Azure Kubernetes Service (AKS) for deployment, Azure Container Registry (ACR)

# Software Architecture

**Web Application (informedinnovations\_web)**

The web application was built using {web application details}.

It is split into the following modules:

|  |  |
| --- | --- |
| **Name** | **Description** |

It is split into the following routes:

|  |  |
| --- | --- |
| **Name** | **Description** |

The following components were used:

|  |  |
| --- | --- |
| **Name** | **Description** |

**Web Services (informedgreetings-services)**

The core of the application was defined using a JDL (JHipster Domain Language) file. This file defines the layout of the application, microservices, and database. The following is a snippet from the file:

As web requests are received, they are filtered, validated, and matched using Spring routing mechanics. The requested API endpoint is then routed to the appropriate method for implementation. The parameters are extracted and specific logic for the desired functionality executed, then a response DTO is returned. The DTOs are then transformed into JSON and returned as a Response Entity.

The AMS WebTools service uses a JSON DTO request object that transforms the DTO object into an XML formatted request to send to the USPS WebTools API to validate an address. XML is returned from the WebTools API and is converted into a JSON response. Note, WebTools can only validate a maximum of 5 addresses at time. For large CSV uploads, it may take a few seconds to validate and create a greeting.

All microservices are running in a Spring Boot application. This application gets packaged into a Docker Container using JIB and stored in an Azure Container Registry. It is then deployed to an Azure Kubernetes Service cluster.

**Authentication**

This backend services and web application use OAuth2 for authenticating and authorizing users. The services are configured using standard Spring Security:  
  
spring:

security:

oauth2:

client:

provider:

oidc:

issuer-uri: https://dev-auth.uspsinnovation.techauth/realms/informed

registration:

oidc:

client-id: ig\_app

client-secret: ig\_app

scope: openid,profile,mailer,email

The website requests an authorization token from the same authentication service using the react-oidc-context and oidc-client-ts libraries. These libraries automatically refreshes access and refresh tokens.

The development authentication server is a Keycloak server running in a Kubernetes cluster in Azure:

Graphical user interface, application

Description automatically generated

**Informed Innovations Keycloak Informed Greetings Client Settings**

Graphical user interface, application, Teams

Description automatically generated

**Informed Greetings / Informed Innovations Login Theme**

# Hardware Architecture

All inbound and outbound traffic goes through a load balancer. Inbound traffic is routed to an NGINX ingress controller on each Kubernetes node. This provides SSL passthrough or termination, URL rewriting, request limits, etc. The ingress controller then routes the request to a registered Kubernetes service. The service points at a pool of available endpoints providing that service.

For the current development purposes, only 1 instance of each container is providing a given service. For a PROD environment, this should be changed to at least 2 for failover purposes, and more if additional systems are available and horizontal scaling is necessary.

Load is automatically balanced across available nodes. A failed container will automatically restart on any available cluster. When deploying a new image (or changing settings), a new container will start up and run a liveness probe to verify successful startup. Once that probe passes, the service will point all new traffic to the new container(s) and start terminating the old containers.

### Load Balancer

An Azure Load Balancer is configured as part of the Azure Kubernetes Service (AKS). The load balancer balances all access to the agent nodes in the cluster.

{TODO: Create an accurate architecture diagram for Small Business Portal/get one from Joel if he already has one}

Diagram

Description automatically generated

**System Architecture Diagram**

# Procedures

### Cluster Configuration Procedure

The Azure Kubernetes Service (AKS) cluster configuration is contained within a Git repository located at git@ssh.dev.azure.com:v3/postaldirect/informed\_innovations/informedgreetings-kubernetes.

The repository is divided into environment folders, dev and prod. Each folder contains a README.md that is in the markdown format. These documents contain all the steps required to set up an installation of Informed Greetings in Azure.

To summarize the install process:

* Create a container registry
* Create a PostgreSQL database
* Create a Kubernetes Instance
* Create the Kubernetes Ingress Controller
* Create the Cluster Issuer
* Create a Config Map for services
* Create Azure Storage
* Create Persistent Volume Claim
* Build and deploy MDR Client container
* Build and deploy Services container
* Build and deploy Web container
* Build and deploy HC PDF container
* Update OPTiMO / NPF Mail Relay
* Install Ingress
* Update DNS configuration

After the install is completed, the AKS cluster will look similar to the following:

➜ informedgreetings (uspsigdev) kubectl get pods -n ingress

NAME READY STATUS RESTARTS AGE

cert-manager-68bfc85c96-wjf4p 1/1 Running 3 64d

cert-manager-cainjector-854fdc4d49-pfc44 1/1 Running 10 64d

cert-manager-webhook-557cd78fb8-28x46 1/1 Running 0 64d

nginx-ingress-ingress-nginx-controller-6878b74896-trc7r 1/1 Running 0 64d

nginx-ingress-ingress-nginx-controller-6878b74896-x4n4j 1/1 Running 0 64d

**Note:** The ingress controller is based on nginx and lives in a separate name.

➜ informedgreetings (uspsigdev) kubectl get pods

NAME READY STATUS RESTARTS AGE

informedgreetings-mdr-6969477559-txt6n 1/1 Running 36 36d

informedgreetings-services-7d8b4d9564-n55zw 1/1 Running 0 2d1h

informedgreetings-web-d4c889fdd-cw4k6 1/1 Running 0 20h

informedgreetings-hc-pdf-server-57ccd46878-jwnkg 1/1 Running 0 120d

To retrieve logs for any of the servers, use kubectl logs, ex:

➜ informedgreetings (uspsigdev) informedgreetings (uspsigdev) kubectl logs -f informedgreetings-services-7d8b4d9564-n55zw

### Application Server Setup Procedure

No additional setup is required.

### DNS Settings

The DNS entries for Informed Coupons are located in the Azure Portal. A record for the ingress should be created that points to the IP address of the ingress for Kubernetes cluster. The services and web sites should be aliased to the ingress:

Graphical user interface, application

Description automatically generated

Note: For prod, the root should also point to production ingress. The ingress will automatically rewrite the address with www.

### Web Server Customization Procedure

All configuration values are in the embedded Tomcat instance via Spring Boot.

### Update Procedure

The CI/CD pipeline handles all deployment tasks.

# Troubleshooting and Reporting

**Q**: How can I prevent robots from indexing development website:

**A**: Add the following to the ingress controller:

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: informedgreetings-ingress

annotations:

kubernetes.io/ingress.class: "nginx"

nginx.ingress.kubernetes.io/rewrite-target: /$1

nginx.ingress.kubernetes.io/use-regex: "true"

nginx.ingress.kubernetes.io/proxy-body-size: "20m"

nginx.ingress.kubernetes.io/proxy-buffer-size: "128k"

**nginx.ingress.kubernetes.io/server-snippet: |-**

**add\_header X-Robots-Tag "noindex, nofollow";**

nginx.org/client-max-body-size: "20m"

cert-manager.io/cluster-issuer: letsencrypt

This will add response headers to all requests. Respectable search engines will honor this header.