FMI Dev Handbook

Developer Onboarding

Hi and welcome to your new life as a Software Engineer at Bidvest Life / FMI.

Getting Started As An FMI Developer

Accounts Setup

First things first, you need to get your `FMI email address` set up and added to our organisation on the various platforms. MS Azure, Gitlab, Sendgrid, Datadog, Sentry and OneSignal.

Speak to @Jaco in the Dev team to make sure this is all set up and ready to go for you.

Authenticator App

You will need to install the MS Authenticator app (or similar) on your mobile phone in order to set up your FMI Azure Active Directory account, as well as to register for and access repositories on Gitlab.

Search your mobile phone's App Store for the `Authenticator` app.

VPN Access

This will be required when trying to connect to `staging` or `production` resources.
You will need to install the FortiClient VPN tool and get the configuration settings from DevOps / SysAdmin

Developer Tooling

As a Bidvest Life / FMI dev you'll need to be able to access and work on various projects, across a number of languages / frameworks and platforms.

Platforms

Developer preference - Windows, macOS, Linux are all supported.

Our systems are all cross-platform, with the exception the http://www.fmi.co.za website, which is built using `Umbraco CMS` and can only be run in a Windows environment.

Frontend

JavaScript code is written according to ES6 conventions, and in some cases using TypeScript.
You will be required to have a number of packages installed on your local environment in order to run the projects.

Package Managers

`NPM` and `Yarn`.

If running macOS, then we would recommend installing packages using the HomeBrew package manager to manage installations and dependencies at an OS level.

Frameworks and CLI's

You should have all of the following installed on your localhost.

Node.js
Angular
Create-React-App
Vue.js - /this is only used for one tiny component in the IDServer project on the login page. Not necessary to install./

IDE / Editor

We recommend that VS Code be used for all front-end development, and that the ESLint and Prettifier plugins are used to format the code and use consistent conventions.

Plugins:
* ESLint
* Prettier
* Azure Account
* Kubernetes

Postman 2

For testing and developing APIs.
Once you are added to the FMI Workspace, you will have access to all the Postman collections that have been created.
This is a critical part of the development process and the ongoing maintenance of these endpoints is essential.

Backend

All our backend services are written using Microsoft .Net Core, and you therefore need to download and install the .NET 5.0 framework and and IDE that is able to support development of .NET applications.

.NET

IDE

Visual Studio Professional on Windows or macOS, or Jetbrains Rider IDE on macOS.
Alternatively, you could use VS Code with plugins on either platform for a light-weight solution.

Databases

- SQL Client - SQL Server Management Studio or Azure Data Studio on macOS
- MongoDB client - Compass

Source Control

All our repos are hosted inside Gitlab, with the exception of one project which houses our Helm template files which is over at Github (for some reason unknown to me, but Jaco has more details on that)

Gitlab

You will need to create your own Gitlab account, and then be added to the FMI organisation and given access permissions by the admin.
Repositories are sorted into logical containers which defines the taxonomy of the project and how it fits into the organisation.

EG:
`fmi-dev-on-premise` `fmi-legacy` `fmi-dev/infrastructure` `fmi-dev/integrations/quotes-integration`

Git Clients

Highly recommend using Gitkraken Free Git GUI for Windows, Mac, Linux | GitKraken for a visual Git experience, but you are free to choose whatever you are most comfortable with. VSCode has all the git capabilities built right in and there are some good plugins for it as well, like Gitlens or Gitgraph.

Third Party Tooling

Naturally, we're rely on a number of third party services to add value to our products. Typically these are paid-for services and we always try to keep the costs as low as possible, so use logins may be shared in some instances.

Google Analytics

Ask Product Owner for access to this, should you need it.

Sentry

This is wired into the DCA (Digital Client Application) so we can record a user's session. Very helpful to see the flow the user took and to debug any issues they may have had in filling out the application form.

User access to this is a shared login - ask the Product Owner

OneSignal

Manages `push notifications` for the FMI Mobile App.
Access needs to be granted by the Product Owner.

Datadog

Logging and telemetry is handled by DataDog and most of our services are wired up to push logging data there.
You will need to be given access to the FMI DataDog account,

The most useful feature there to debug issues will be the `Logs > Search` function, which can be filtered to tail live data, or run details searches.

Apple Developer / TestFlight

Refer to the project ReadMe for more details about getting this set up.

Infrastructure

We run a micro services architecture deployed to a `Kubernetes Cluster` hosted inside `MS Azure`,
`Consul` from HashiCorp is the mechanism which facilitates secure `Service-To-Serice` networking inside the cluster.
A future enhancement will include us moving over to a Service Mesh.

Microservices

We try to follow the general best practices around a micro services architecture.
Microservice Architecture pattern
Each micro service generally has its own database and is publicly exposed to other services within the cluster through the `Consul` configuration files.

Kubernetes

You can read more about Kubernetes here

Suffice to say, you will need to download and install the K8s tools, including the kubectl command line tool. Head on over here and follow the instructions for your platform of choice.
Install Tools | Kubernetes

Once you have the kubectl CLI installed, you can add the Kubernetes extension to Visual Studio Code by searching for it in the market place.
You will need the extension to set the current cluster and enable port forwarding, when developing locally.

Consul

Networking inside the cluster is facilitated via Consul, which is said to be a “chatty” protocol. IE it is what allows all the services to register and communicate with each other.

You will need to install Consul locally, and then port forward to the Consul Server inside the K8s cluster. See below for that process.

Setting up and running Consul
Install a Consul Agent on your local machine

Port Forwarding

While developing locally, you will need to communicate with services inside the K8s cluster.
To do that, you need to port forward all traffic on a port to the cluster.

As of right now, the command to do that is as follows, but please note that the `Pod` locations and ports could change in the future, although it is rare that this happens in practice.

```
kubectl port-forward pods/consul-server-2 8500:8500 -n consul
```

The above command will forward all local traffic on port 8500 to your currently selected/active K8s cluster (as configured with kubectl) to the locations `pods/consul-server-2:8500`

Please read the below link for more information regarding this process.
Use Port Forwarding to Access Applications in a Cluster

Development Processes

Once all your tooling has been set up, you are now ready to begin working on a project.

We use Atlassian Jira and Confluence to manage our development workloads and roadmap.
Grooming and analysis are all done according to typical agile practices.

Feature Development / Bug Fixes

When it's time to build out a feature, or fix a bug, you will typically need to do the following:

1. Read the ticket inside JIRA and ensure that it is clear what needs to be done. NB that it's important for the `Acceptance Criteria` to be defined.
2. Using your preferred Git client, pull down the repository to your local host and ensure that it is up to date with source.
3. Create a new branch, using the naming convention `feature/<ticketnumber>` `bug/<ticketnumber>` `fix/<ticketnumber>`
4. Upon creating the new branch, you should immediately push it to `origin` and sync any commits as you go along. This will ensure that progress can be tracked and work will not be lost.
5. Once your work is complete and you have tested it locally, open a new `Merge Request`. A team member or team lead will then need to approve the MR, after which it must be merged into the `master/main` branch
6. Your code will now be run through the Gitlab CI/CD process, and may be required to pass a number of tests. Any issues discovered int his phase will need to be resolved and the process repeated.
7. Only once your code has been merged into `master/main` and all tests have passed, can the ticket move to a "testing" phase, where it will be measured against the Acceptance Criteria that was defined in the ticket.
8. After is passed this phase, your feature will be ready to be deployed.

Deployments

These need to be carefully managed and coordinated with the rest of the team.
Releases should be tagged and release notes should be added to the Gitlab release log.

We're attempted to follow the Gitlab flow, but this process could use some refinement. The end goal should be continuous deployment and releases.

The Projects

Here is a brief overview of the most common and important projects that you are likely to work with at the beginning.
More details on these various components can be found in Confluence, but here's what you need to know as a new Dev starting out and you should probably sync these repos down to your localhost first up.

Maverick Azure Data (MAD API)

https://gitlab.com/fmi-dev/maverick-azure-data
.Net Core project

This is an API interface to pull data out of a very large SQL database.
This is where we source all our information from that pertains to Adviser Details, Client Details, Policy Information and so forth, and it is synced from an on premise DB which drives all our legacy systems.

DCA (Digital Client Application)

React.js using Next,js framework and Express.js middleware.
Auth is handled using Passport.js

The UI component of our new Client Application system and workflow.
Facilitates the process whereby a client or adviser will fill in all the details needed around an application to get covered.

It's seeded with data from the legacy Quotes Package

Instructions for running the project are contained in the readme

Application

Repo

Workflow Integration

Repo

Aura Integration

Repo

Communication Hub

Repo

Communication Hub Packages

Repo

Adviser Portal

Repo

Brokerage Onboarding

Repo

FMI.co.za

#FMI