Patrik Pfaffenbauer

Portfolio

patrik.pfaffenbauer@p3-software.eu | +43 660 49 08 028 | Software Engineer
Untere Dorfstraße 14, 4622 Eggendorf | AUSTRIA | ATU73971525

Patrik Pfaffenbauer
26.01.2024

Contents

About me

My name is Patrik Pfaffenbauer, I live in Upper Austria and work as a self-employed Software Engineer/Architect.

You can find some information about my technology stack, projects I have made and some more stuff. But let me start with my favorite quote.

Quote

If you think good architecture is expensive, try bad architecture.

Brian Foote & Joseph Yoder

Technology Stack

Programming Languages

- C#
- JavaScript/Typescript
- C/C++
  - + Embedded micro controller
- Unix Bash

Technologies

- .NET Core
- Angular
- Qt
- docker & docker-compose
- Terraform & Ansible
- Blockchains
  - Bitcoin
  - Ethereum
  - DeFiChain

Cloud

- Microsoft Azure
- Amazon Web Services
- Power Platform

Concepts

- Clean Code
- SOLID principle

Passion

- Bitcoin & DeFi
- Building automation aka smart home

Book recommendations

- **Clean Code/Architecture/Coder/Agile** by Robert C. Martin
- **Lean Startup** by Eric Ries
- **Mastering Bitcoin** by Andreas Antonopoulos
- **The Bitcoin Standard** by Saifedean Ammous

BeKa-Software GmbH

Develops individual software for different customers in Vienna and Pasching. (www.beka-software.at)

SystemTera

This project involves monitoring of around 500 heating plants for multi-party housed, commercial parks, hotels, and industrial companies in the Upper Austria area. The project then became a product. (www.systemtera.com)

Server

We developed a proprietary Linux embedded controller, which run on Gentoo Linux. The application was developed with C++ and Qt. The controller collected data via various bus systems (KNX, ModBus, MBus, …) and transfers them to the “cloud”.

Cloud

The cloud application manages the servers in the field and stores the data in the database using the Entity Framework. This application was developed with C # .NET. Communication between the server and the cloud was implemented with SignalR in order to be able to query real-time data. I developed a SignalR client in C ++ as open source software (https://github.com/p3root/signalr-qt). The servers in the field are managed by the manager.

Manager

The manager is the administrative interface for the entire system. Here new servers could be created and the server data visualized. The application was implemented in WPF. Communication between Manager & Cloud is established via WCF & REST. Communication with the server was realized with SignalR. Various reports on the efficiency of the systems were also created.

Tasks/Technology

- C# .NET
  - WCF
  - WPF
  - REST
  - SignalR
  - Entity Framework
- C++
  - Qt 4.8.2
  - SignalR
- Embedded Linux
  - Gentoo
- SVN

Verbund Meter

Together with the Verbund company, an electricity meter was developed for the Verbund Eco-Home product. (https://www.verbund.com/de-at/privatkunden/smart-home/eco-home-zusatzgeraete/verbund-strommessmodul)

This project runs on an Atmel SAM processor and was implemented with C/C ++. The challenge here was clearly stability and meter accuracy. There is no way to update the electricity meter over the Internet.

The meter provide 3 different interfaces:

- ModBus
  - To provide an open standard, so that 3rd party applications can read data from the meter.
- PLC (Power Line Communication)
  - For the Verbund Eco-Home system.
- USB
  - To calibrate and update the meter.

Tasks/Technologies

- Embedded
- C/C++
- Git & Bitbucket

NETxAutomation Software GmbH

Is one of the leading providers of software in building automation. (www.netxautomation.com)

Project time

2016-12 – 2019-02

Web Manager

Is the configuration interface where customers can monitor and configure the building. This includes alarming, scheduling, trending and visualization. This application was developed from scratch with Angular2 **under my responsibilities**. Angular 7 is now used here. Communication takes place via REST & SignalR. The backend is developed in C and uses a COM interface for data transmission to the server.

Server

Is the **core** of the product range, this collects the data from different bus systems and distributes them depending on the configuration. The server is implemented in C ++.

Plugins

The system can be expanded using a plug-in based system. This can be done via C++ and .NET. New drivers where only written in .Net.

For this I haved developed some “drivers”:

- LaMPs
  - To bring different DALI KNX gateways to one common denominator
- KNX IP Secure
- MBus UDP

Vision

NETxVision is the name of the mobile app that is used for the visualization. This was developed using Xamarin.

Tasks/Technologies

- Teamleiter der Softwareentwicklung
- C# .NET >= 4
- C# .NET Core >= 2.1 (Reporting)
- C++
- Angular >= 2 + Typescript
- Git / Bitbucket
- Xamarin

AutoLogg GmbH

AMV Networks GmbH are working on solutions for the mobility of tomorrow. For this I have participated in 2 projects.

Project time

2019-02 – to date

AutoLogg

AutoLogg is a digital driver logbook, which also complies with the Austrian/German tax administrations. To do this, the customer must install the AutoLogg Box/Dongle in their vehicle and can use it to log his tracks digital. However, since new vehicles area already delivered with onboard connectivity, installation of the box has become obsolete. (www.autologg.com)

AutoLogg Connected

Tesla offers the possibility to read vehicle data in real-time. For this purpose, a .NET core application was developed that connects to Tesla and reads the vehicle data. As soon as a trip is recognized, it is transferred to the AutoLogg system.

AutoLogg CRM (Power Platform)

AutoLogg depends on several systems. We wanted to provide all those information in one place and started implementing a CRM using Microsoft Power Platform.

We need to sync data from several systems:

- WordPress (Shop)
  - Subscription handling, license check, ..
- Backend
  - Information about vehicles
- Connected
  - States of the connected vehicles
- Jira
  - Customer success
- ….

Tasks/Technology

- AWS ECS
- AWS DynamoDB
- AWS Lambda
- AWS API Gateway
- AWS SQS
- AWS SNS
- AWS Cloud Watch
- AWS … many more
- Azure DevOps
- Reporting
- MS Azure

- App Services
- Microsoft PowerApps
- OData
- Docker
- Keycloak
- .NET Core >= 5.0
- Frontend
    - Angular >= 2
- Autologg Backend
    - Java
- IaC – Terraform
- Power Platform
    - Power Apps
    - Power Automate
    - Power BI

Arolla.io

Arolla.io is an E-Scooter Sharing Platform. (www.arolla.io)

I was responsible for Backend and Frontend.

Tasks/Technologies

- Payment & Vouchers
    - React Native App
    - Heidelpay (Payment Provider)
    - AWS Lambda
        - NodeJS
        - Python
    - GraphQL
    - AWS Cloudformation

Insurance Industry

Project time

2021-01 – 2022-06

Power Platform

Managing contracts, claims, customers.
Providing APIs for several 3rd parties to create and update contracts, customers, etc.
AI assistant fraud protection.
Automated charging for the insurance contracts (pay-as-you-go).

Technologies

- Power Platform
  - Power Apps
  - Power Automate
- Azure DevOps for CI/CD
- Azure
- App Services
- API Management
- Azure WAF
- Azure Data Lake
  - Dataverse Connector
  - Analysis and reporting
- Docker
- Sharepoint
- IaC - Terraform

P3bble

Is a framework to communicate with the Pebble Smartwatch. This was developed for Windows Phone 8 (C # .NET). I stopped working on it because Microsoft has stopped Windows Phone as well. (https://github.com/p3root/p3bble)

Automatica.Core

Automatica is a self-developed automation system. The server is developed with .NET Core and can therefore be run on Windows, Linux and Mac. The configuration interface is developed with Angular and is delivered on the web server operated on the host. (https://www.automaticacore.com, https://docu.automaticacore.com)

Project time

2017-01 – to date

Server

The server is the central piece, this takes care of the interfaces to the outside (Web API, REST & SignalR for Realtime Date) and loads the various plugins. This also processes the data from the plugins (drivers and logics) and dispatches them on to the next plugin, depending on the configuration.

Visualization

The visualization is generated from the configuration properties.

Drivers

I already developed a wide set of drivers:

- KNX IP (+ IP Secure)
- MBus UDP
- ModBus TCP / RTU
- Consants
- EnOcean
- Fronius Symo Inverter (Solar API)
- Amazon Alexa, Logitech Harmony
- Apple HomeKit
- Ikea Tradfri
- Loxone Miniserver (WebSocket API)
- OpenWeatherMap
- Times/Sun
- Wake on Lan
- ZWave (Work in Progress)
- ZigBee (Work in Progress)

Logics

I already developed a wide range of logics:

- Messenger
  - Send emails
- Compare
  - Bigger, Equals,…
- Logic

    - And, Or,…
- Math
    - Addition,…
- Time
    - Zeitschaltuhr, …

Cloud

The cloud application is also developed in .NET Core and is hosted in Azure. This application is currently only used for license management, extension/update management. There is also a management interface for this, which is implemented in Angular.

CLI

The CLI (Command Line Interface) is a small auxiliary tool for the development of plugins. This can e.g. generate a plug-in boiler plate where all dependencies and basic implementations already exist. With this CLI, a plugin can also be built and deployed to the cloud.

CI/CD

We use Azure DevOps for build and release management. There are various build / release pipelines. Good test coverage was important to me here, an error in the core makes the system unusable. Therefore, I have a test coverage of> = 70%.

Extensions

Each plugin has its own pipeline, which is generated by Automatica.CLI when the project is created. After each build, the packaged plugin is automatically transferred to the dev cloud.

Raspberry PI Image

An image is also generated for the Raspberry PI in Azure DevOps which only needs to be copied on an SD card and the user is ready-to-go.

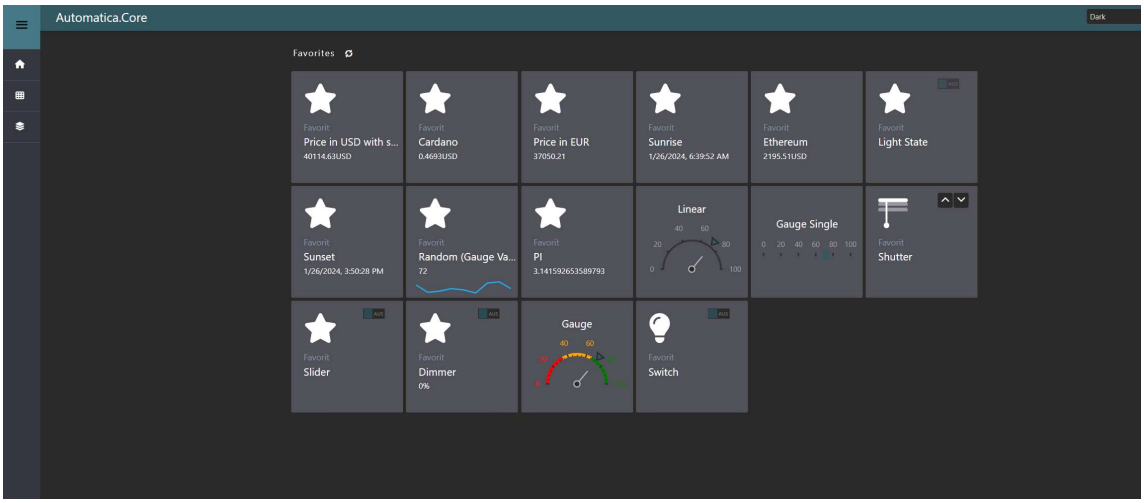
Tasks/Technologies

- C# .NET
    - Entity Framework Core
    - SignalR
    - Web API (REST)
- Angular >= 2
    - DevExtreme UI Components
- Embedded Linux
    - Raspberry PI (Debian)
- DocFX
    - Für die Code Dokumentation
- Azure Cloud
- Azure DevOps

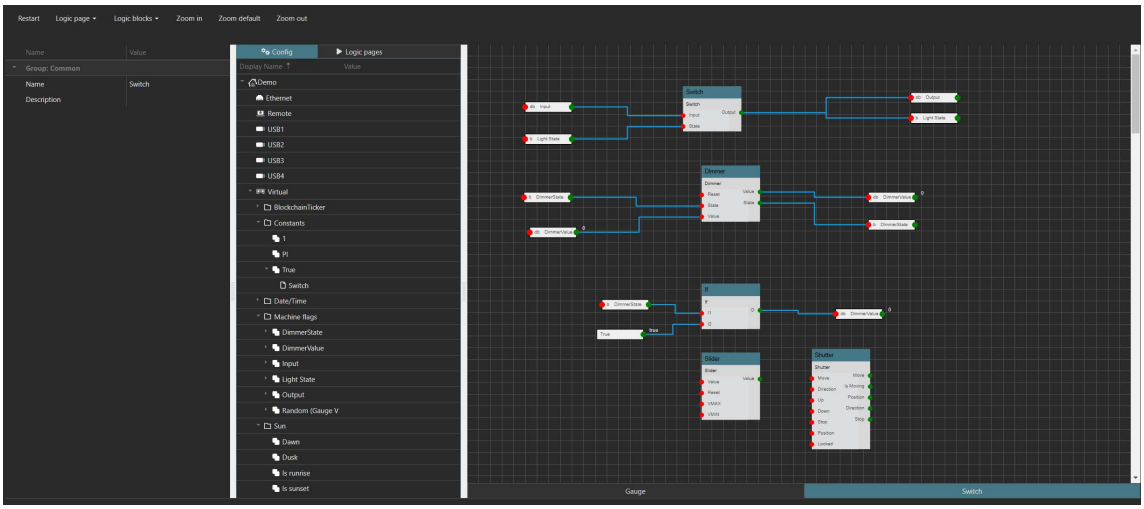
- GitHub
- Git
- Docker
- Terraform
- Crypto/Hashing
    - Secure Remote Passwort
    - HDKF-SHA512
    - ED25519 Key Exchange
    - ChaCha20-Poly1305 AEAD
    - Curve25519
    - ED25519
    - AES128
    - PBKDF2

Screenshots

Visualization



Programing Interface



Links

Look here to get some more information.

https://github.com/automatica-core

https://www.automaticacore.com

https://demo.automaticacore.com

saiive.live DeFi Wallet

„saiive.live“ is a lite wallet app for the DeFiChain Blockchain (https://www.defichain.com) and Bitcoin. Written in Flutter and runs on Android, iOS, Windows, Linux, Ubuntu, and so on.

The backend infrastructure is hosted in Azure and provides a generalized service for several blockchains.

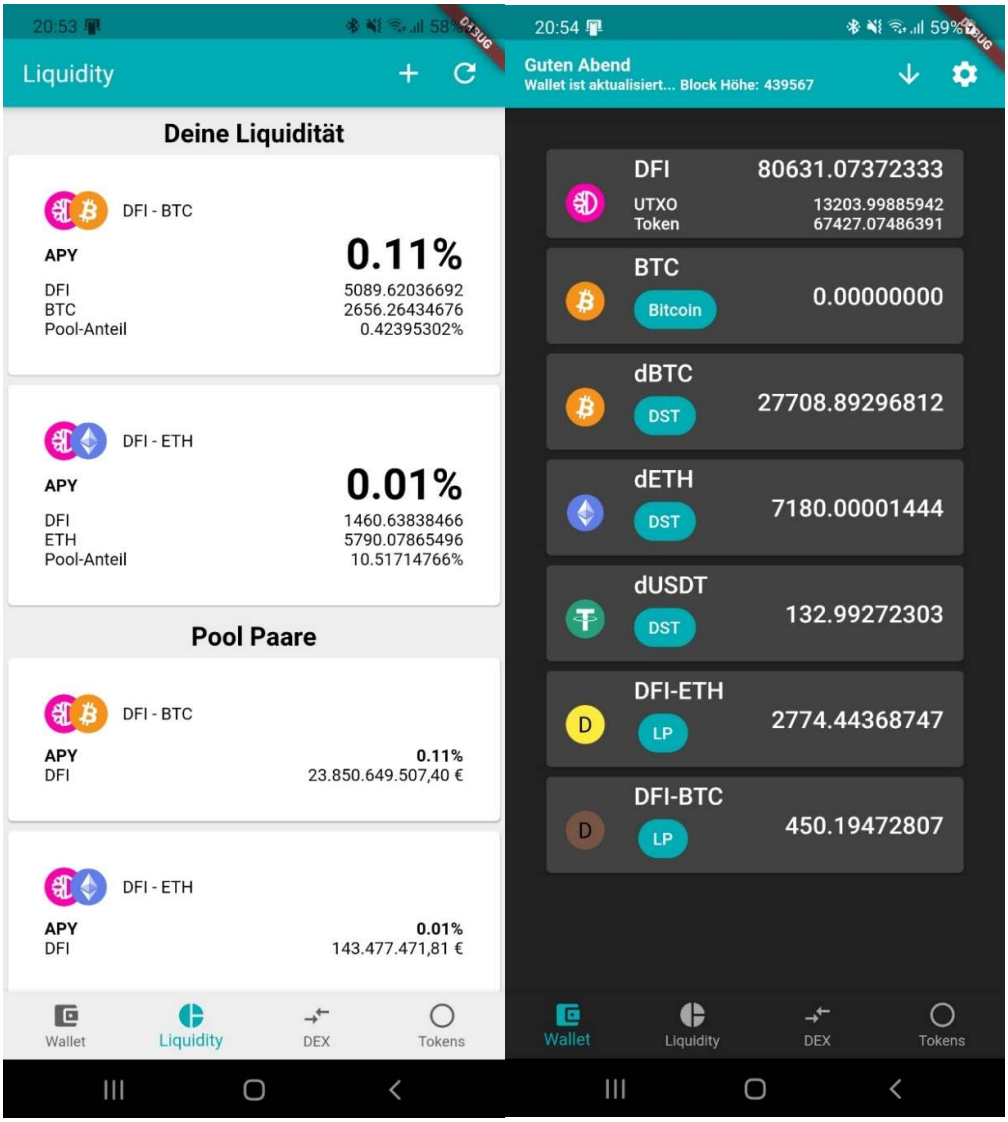
DeFiChain

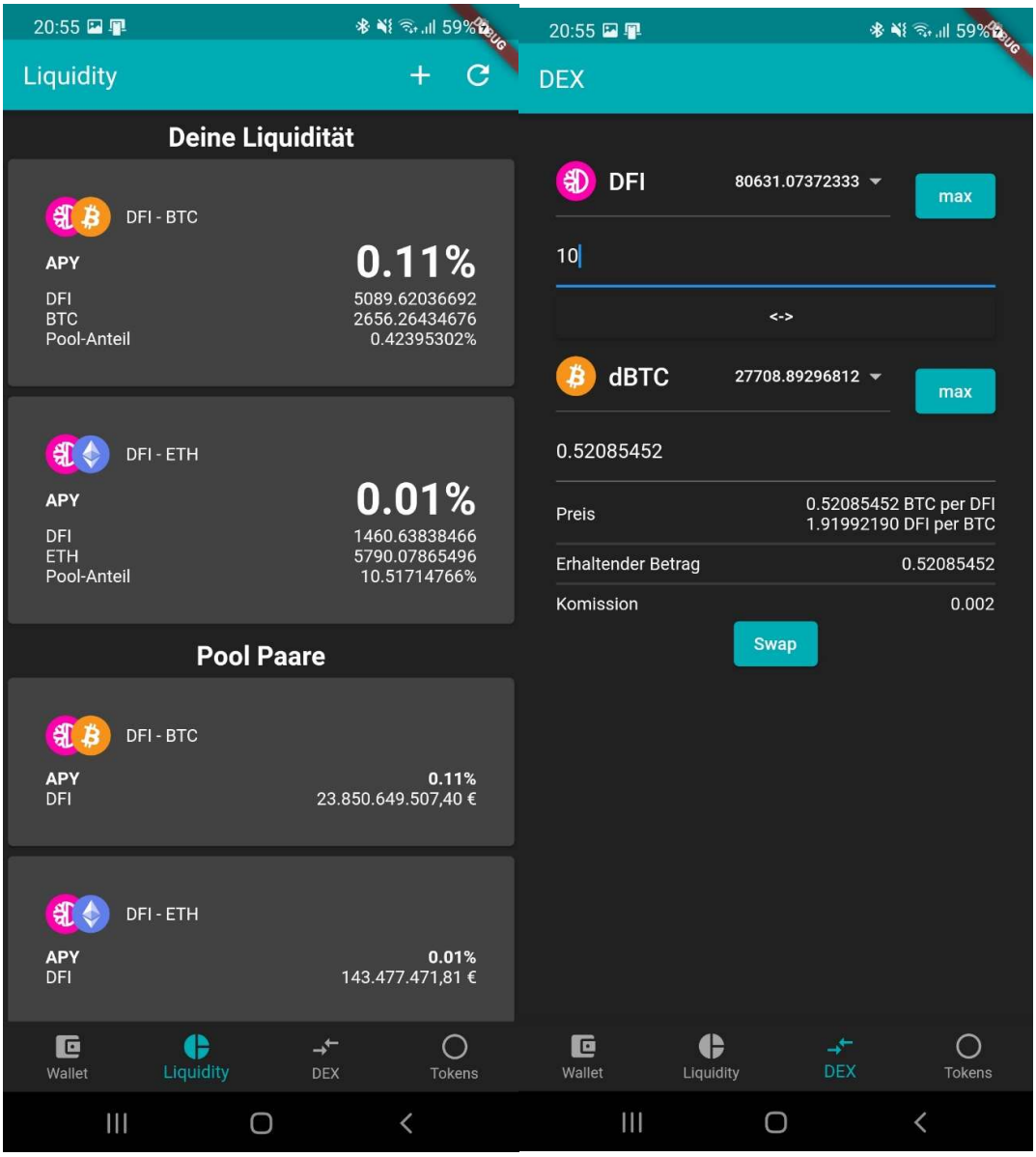
DeFiChain is a Bitcoin fork with additional DeFi functionalities. I have reverse engineered the DeFi custom transactions and built open source dart library to create such custom transactions.

Open Source

The wallet is complete open source and can be found on GitHub (https://www.github.com/saiive & https://www.github.com/saiive/saiive.live).

Screenshots





Unfortunately we needed to abandon this project cause lack of funds. Code is still open-sourced and other community members decided to continue the project.

Links

Homepage: https://www.saiive.live

GitHub: https://www.github.com/saiive

API: https://supernode.saiive.live/api/swagger/ui