# Introduction

## Table of Contents

# Motivation

## Motivation and Goal

In the last years, we built several process applications and process platforms on behalf of the customers using Camunda BPM Engine. In doing so, we were able to extract common requirements, especially if the task-oriented application has been implemented. The requirements were basic and independent of the frontend implementation technology. It turned out that some issues occurred every time during the implementation.

These were:

- coping with performance issues of Camunda BPM Engine by the big amount of tasks available (total tasks, tasks per user)
- creating high-performance custom queries for pre-loading process variables for tasks
- creating high-performance custom queries to pre-load business data associated with the running process instance
- high-performance re-ordering (sorting) of user tasks
- high-performance retrieving of tasks from several process engines
- repetitive queries with same result
- creating an archive view for business data items handled during the process execution
- creating an audit log of changes performed on business data items

Since we were developing customer software, we found solutions to those requirements and gathered experience in applying different approaches for that. Some issues listed above result from fact that data on a single user task is being read much more often than written, depending on the user count. For systems with a big amount of users this becomes a serious performance issue and needs to be addressed.

A possible solution to most of those issues listed above is to create a special component which has a read-optimized representation of tasks. Such component acts as a cache for tasks and allows for serving a high amount of queries without any performance impact to the process engine itself at the costs of loosing strong consistency (and working with eventual-consistent task list).

We successfully applied this approach by multiple customers and identified the high initial invest as a main drawback of the solution. The goal of this project is to provide a component as a free and open source library, to be used as a foundation for creation of process platforms for Camunda BPM engine. It can also be used as an integration layer for custom process applications, custom user task lists and other components of process automation solutions.

# Features

## Features

### Supported process engines

- Camunda BPM

### Task List

A task list is an application allowing to represent a list of user tasks. This list is created based on user's profile (including authorizations based on roles) for every user. The following features are provided by the library:

- user task API providing attributes important for processing
- mirroring tasks: provides a list of tasks in the system including all task attributes provided by Camunda BPM Engine
- reacts on all task life cycle events fired by the process engine
- high performance queries: creates of read-optimized projections including task-, process- and business data
- centralized task list: running several Camunda BPM Engines in several applications is a common use case for larger companies. From the user's perspective, it is not feasible to login to several task lists and check for relevant user tasks. The demand for the centralized task list arises and can be addressed by the the use of the `taskpool` library if the tasks from several process engines are collected and transmitted over the network.
- data enrichment: all scenarios in which the data is not stored in the process payload result in a cascade of queries executed after the task fetch. In contrast to that, the usage of the `taskpool` libary with a data enrichment plugin mechanism (allowing to plug-in some data enricher on task creation) allows for caching the additional business data along with the task information, instead of querying it during task fetch.

### Archive List

An archive list provides a list of business objects processed during the execution of business process. Such a business object lifecycle spans over a longer period of time than the process instance - a common requirement is to get a list of such objects with different statuses like preliminary, in process or completed. The `datapool` library provides the following features:

- business object API providing attributes important for processing
- business object modification API for creating an audit Log
- authorization API for business objects

### Process List

A process list provides a list of running instances and a list of process definitions deployed in the process engines connected to the library.

- list of startable process definitions (including URLs to start forms)
- list of running process instances
- reacts on life cycle events of process instance

# Solution Architecture

## General Idea

The implementation of a single (small) process application can be performed mostly using the Camunda BPM library itself. If the solution becomes larger, for example by setting up multiple engines for different processes or the load on a single process engine becomes unmanageble for a process engine cluster, it is worth to separate the solution into *process specific* and *process agnostic* parts. We call the *process specific* part of the solution Process Application and the *process agnostic* part - Process Platform.

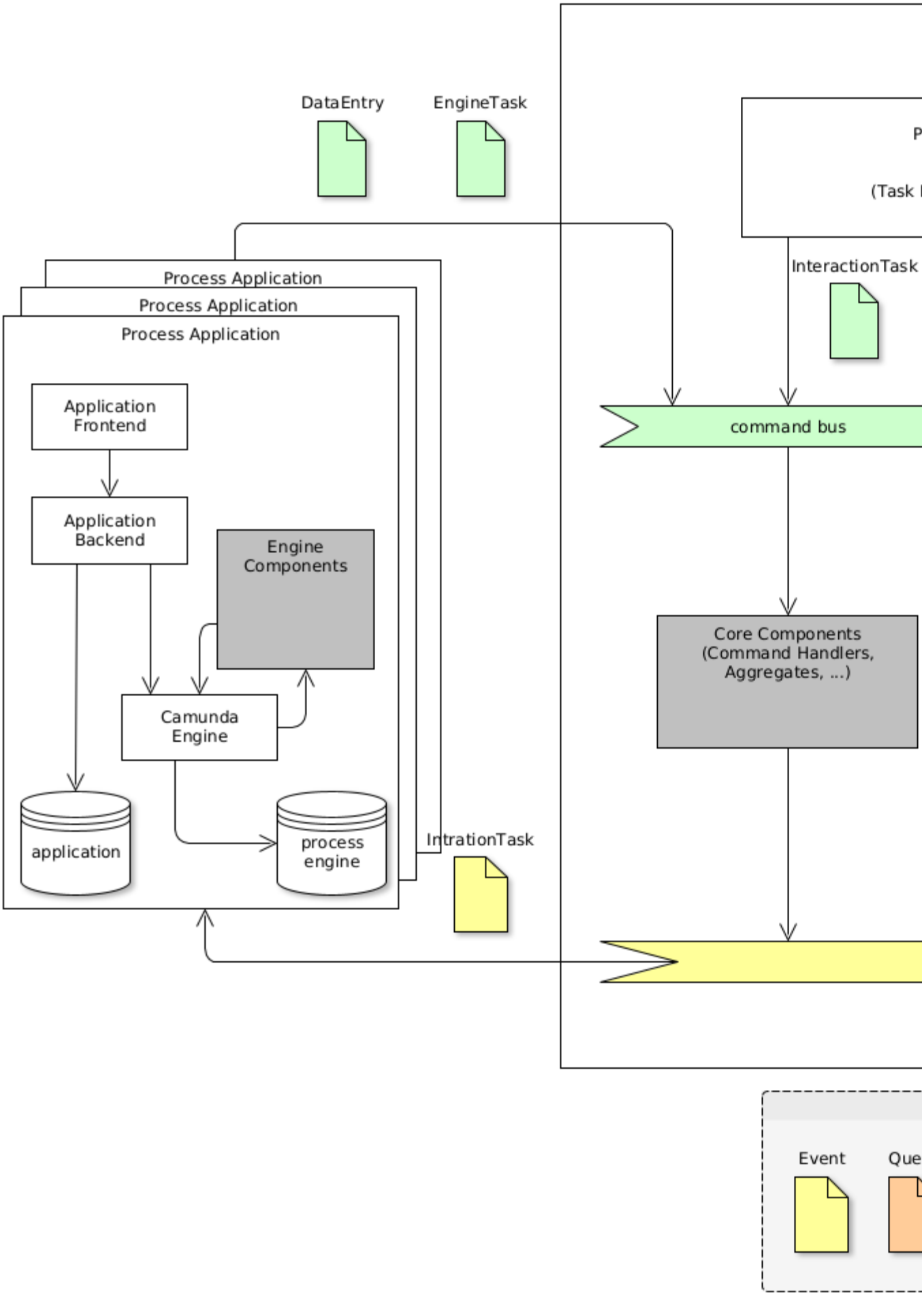
Based on the assumption of the asymmetric read/write characteristics of task-oriented process applications, we decided to apply Command Query Responsibility Segregation (CQRS) pattern during the architectural design. As a result, we supply components to collect the user tasks from the process engines and create a read-optimized projections with user tasks and correlated business data. The components can be easily integrated into process applications and be used as foundation to build parts of the process platform.

## Design Decisions

We decided to build thi library as a collection of loose-coupled components which can be used during the construction of the process automation solution in different ways, depending on your Usage Scenario.

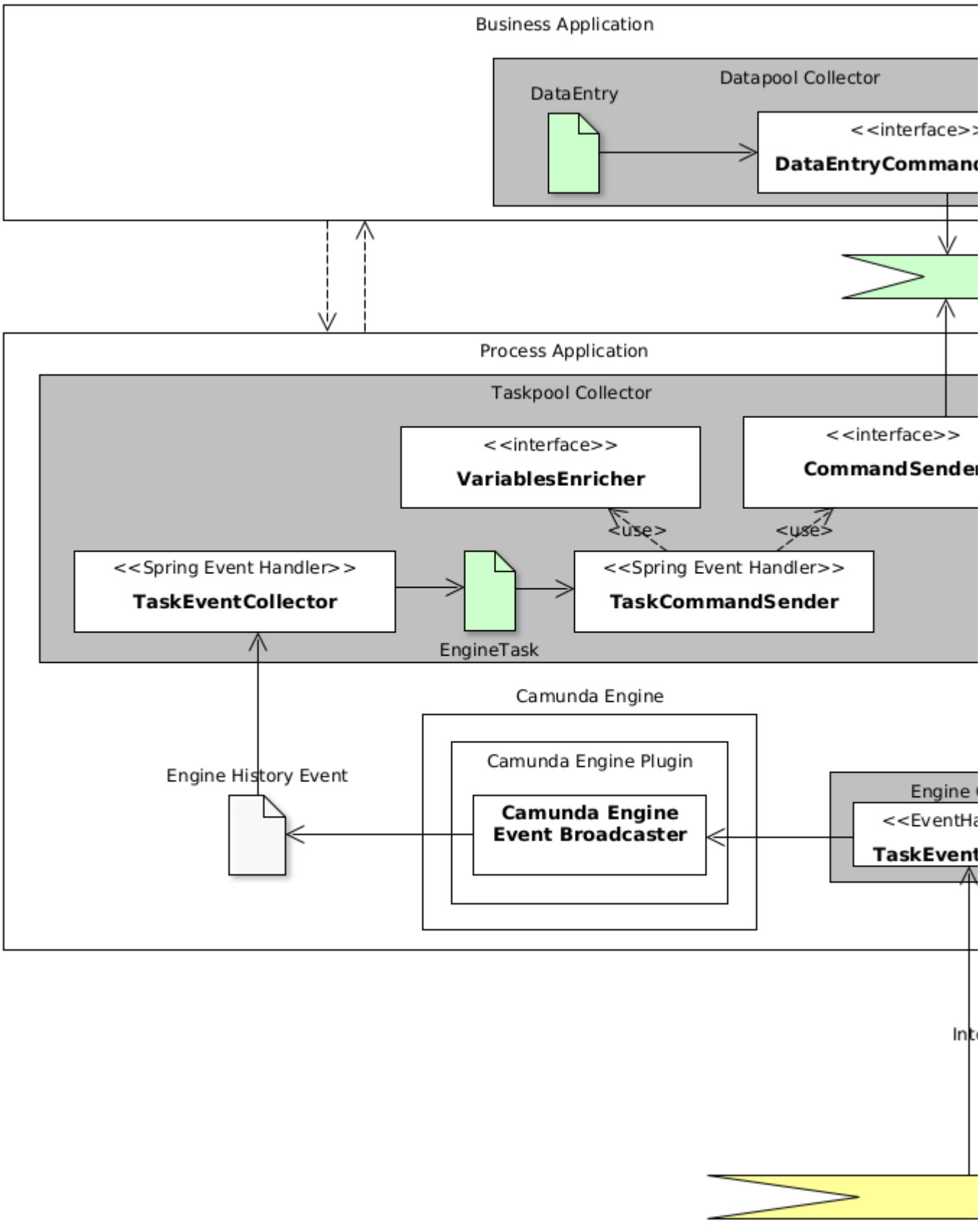
The process platform is a central application consisting of business process independent components like a central user management, task inbox (aka task list), archive view for business data processed, audit logs and others. One or many process applications integrate with the process platform by implementing individual business processes and provide user tasks and business data changes to it. They may also ship application frontends which are integrated into/with the frontends of the process platform, including business object views, user task forms and other required pieces.

The following diagram depicts the overall logical architecture.:




## Implementation Decisions

The components are implemented using Kotlin programming language and relies on SpringBoot as execution environment. They make a massive use of Axon Framework as a basis of the CQRS implementation.



# Further outlook

## Further outlook

This library serves as a foundation of several follow-up projects and tools:

- Skill-based-routing: based on information stored in the taskpool, a skill-based routing for task assignment can be implemented.

- Workload management: apart from the operative task management, the workload management is addressing issues like dynamic task assignment, optimal task distribution, assignment based on presence etc.