**Camunda Platform 8 for .NET Developers**

*In this tutorial, learn how you can take advantage of Camunda for powerful process orchestration as a .NET developer.*

**Table of Contents**

* [**Use case description**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#usecase)
* [**Getting started with Camunda Platform 8**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#gettingstarted)
* [**Building the process application**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#buildingprocess)
  + [**Modeling the process**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#modelingprocess)
  + [**Client implementation**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#implementation)
    - [**Process deployment**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#processdeployment)
    - [**Starting a process instance**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#processinstance)
    - [**Getting some work done**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#workdone)
* [**Camunda Platform 8 Walkthrough**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#camunda8walkthrough)
  + [**Conclusion**](https://camunda.com/blog/2022/11/camunda-platform-8-dotnet-developers/#conclusion)

With the [**Camunda Platform 8**](https://camunda.com/platform/) launch we [**boosted .NET**](https://docs.camunda.io/docs/apis-clients/community-clients/c-sharp/) as part of our [**polyglot initiative**](https://docs.camunda.io/docs/apis-clients/working-with-apis-clients/). This guide steps through creating a process automation application in .NET, leveraging all the potential Camunda Platform 8 has to offer.

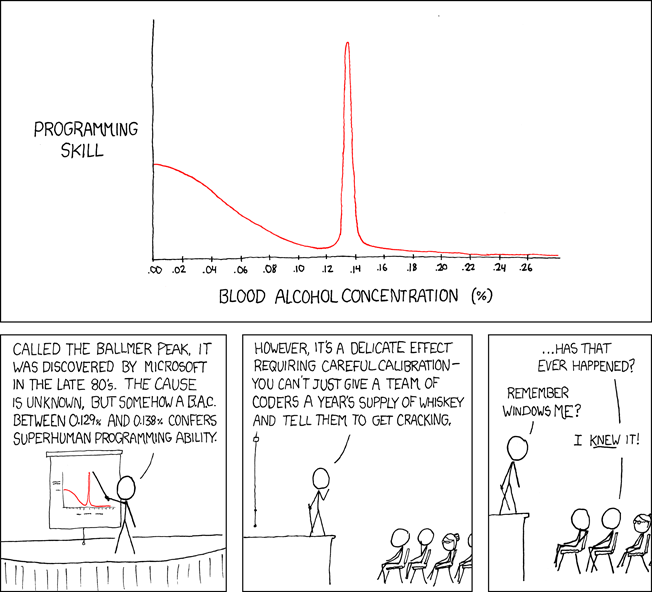
First, we’ll take a look at how we access Camunda Platform 8. After discussing the foundations of the Camunda Platform, we’ll then develop our application by designing our workflow. Next, we’ll investigate the Zeebe C# client and make use of it in our .NET microservice. To conclude this blog post, we’ll summarize the lessons learned.

**Use case description**

Before we begin, let’s align on the application we want to build. In this blog post, we’ll create an application around the famous Ballmer-Peak.

***Disclaimer:****The Ballmer Peak is mythical. This is not a real scientific study, and this effect is not a validated fact. We are using this scenario as a fun example. To be clear: we do not encourage drinking alcohol to improve working ability.*

People say that Steve Ballmer, the former CEO of Microsoft, conducted a study to analyze how blood alcohol concentration (BAC) impacts the programming skill of a developer. Surprisingly, it turned out that developers with a BAC between 0.129 % and 0.138 % showed a superhuman programming ability–the so-called “Ballmer Peak.” The challenge, however, is the careful calibration needed to reach this concentration. If developers drink too much or too little the skill level reaches rock bottom. By choosing this use-case, we have a nice relation to .NET overall since both originated from Microsoft.



This fun illustration shows the findings of this mythical study.

To circle back, the application we want to design should support you in calculating how many alcoholic beverages you need to drink to reach the Ballmer Peak.

The algorithm for this is quite simple and depends on the gender and weight of a person.

After capturing these metrics, we can now familiarize ourselves with Camunda Platform 8.

**Getting started with Camunda Platform 8**

Camunda is a solution for [**process orchestration**](https://camunda.com/process-orchestration/) that allows you to orchestrate complex flows across people, systems, and devices, transform your organization digitally, and automate nearly any process anywhere. Some of the popular use cases include [**human task**](https://camunda.com/solutions/human-workflow/) Orchestration, [**microservice orchestration**](https://camunda.com/solutions/microservices-orchestration/), and the [**modernization of legacy IT systems**](https://camunda.com/solutions/modernize-legacy-it-systems/). I tend to say that whenever you can imagine a process, which is basically the algorithms of a company, you can capture and automate it with Camunda Platform.

To achieve this, multiple tools are provided which contribute to the process automation lifecycle.

* [**Web Modeler**](https://camunda.com/platform/modeler/) is used for designing processes and decisions in the standardized [**BPMN**](https://docs.camunda.io/docs/guides/automating-a-process-using-bpmn/#what-is-bpmn)/[**DMN**](https://docs.camunda.io/docs/components/modeler/dmn/) format so both developers and business stakeholders alike can understand the components of complex business processes. Modeler can be seen as a hub where multiple people can collaborate and discover processes and decisions in the same place.
* [**Connectors**](https://camunda.com/platform/modeler/connectors/) are out-of-the-box components which can be used in the process right away. There are, for example, [**Slack**](https://docs.camunda.io/docs/components/integration-framework/connectors/out-of-the-box-connectors/slack/), [**SendGrid**](https://docs.camunda.io/docs/components/integration-framework/connectors/out-of-the-box-connectors/sendgrid/), and [**REST**](https://docs.camunda.io/docs/components/integration-framework/connectors/out-of-the-box-connectors/rest/) Connectors [**available today**](https://camunda.com/blog/2022/11/increase-process-productivity-google-drive-slack-sendgrid/), with more coming. Users can also create their own reusable components. This blog post features a technical [**sneak peak into the Connector architecture**](https://camunda.com/blog/2022/07/a-technical-sneak-peek-into-camundas-connector-architecture/).
* The **[Zeebe](https://camunda.com/platform/zeebe/)** workflow engine is the powerhouse of Camunda Platform 8. The engine is built to run cloud-native and scale in a linear fashion to support high load scenarios.
* [**Operate**](https://camunda.com/platform/operate/) gives you visibility into the workflow engine. Usually this tool is used by a process operator who checks and controls the lifecycle of process instances and definitions.
* To enable users to work on steps in the process, Camunda features **[Tasklist](https://camunda.com/platform/tasklist/)**.
* To continuously improve the process, [**Optimize**](https://camunda.com/platform/optimize/) is also a part of the package. Optimize allows you to analyze the process for inefficiencies and bottlenecks.

During this blog post you will be exposed to most of these tools, so stay tuned!

Camunda is either available as [**Software-as-a-Service**](https://camunda.com/get-started/) (SaaS) or can be hosted on your [**own premises**](https://docs.camunda.io/docs/self-managed/about-self-managed/). For the sake of simplicity, this blog post focuses on the SaaS version. If you want to try out this process application or build something on your own, you can get started with a [**30-day free trial**](https://camunda.com/get-started/). As you join the journey, take a look at my [**GitHub repository**](https://github.com/Hafflgav/camunda-8-dotnet-guide) and follow along.

**Building the process application**

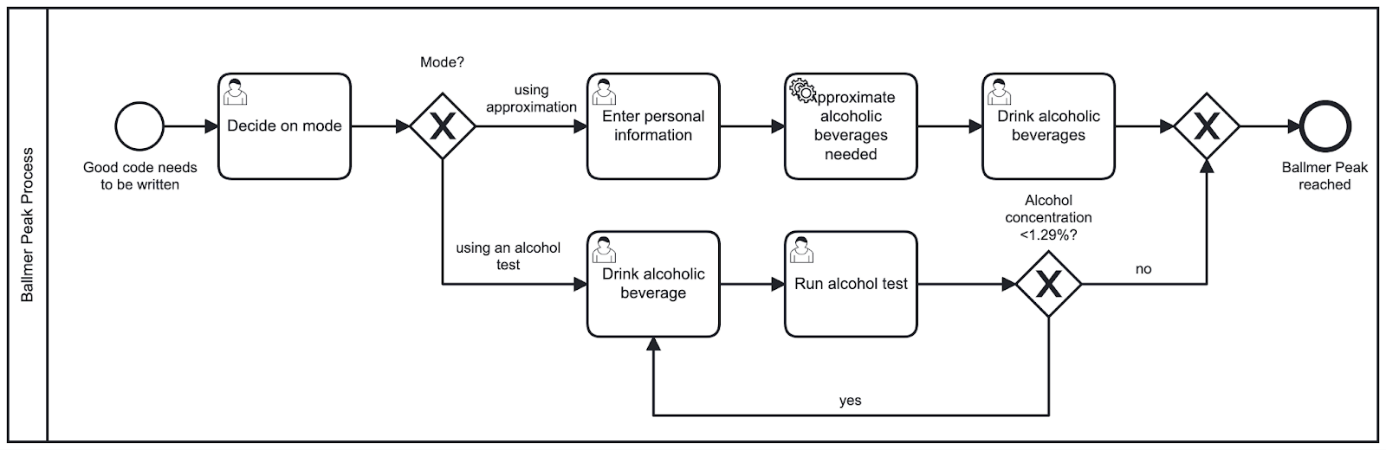
We can separate the process of building an application for the Ballmer Peak in these three steps:

1. Design a process.
2. Implement our client in .NET.
3. Take our system on a test run.

**Modeling the process**

To kick off development, you first need to align on a process. This is accomplished in [**Web Modeler**](https://camunda.com/platform/modeler/). Camunda Platform 8 leverages the [**BPMN 2.0 specification**](https://docs.camunda.io/docs/components/modeler/bpmn/bpmn-primer/). In contrast to other tools that use proprietary notation, Camunda Platform uses this [**standardized**](https://www.omg.org/spec/BPMN/2.0/About-BPMN/) and open notation to capture processes and [**very complex flows**](https://camunda.com/blog/2022/07/why-process-orchestration-needs-advanced-workflow-patterns/). Before we dive into modeling, here are some further specifications:

* A user should either be able to use an approximation or manually perform blood alcohol tests.
* To approximate the BAC we only want to rely on gender and weight.

*Figure 2: Ballmer peak BPMN process*

I developed the BPMN diagram in Figure 2 to represent our process. After starting an instance of the process, a user can choose if they want to use the approximation or manual alcohol test.

Following the top path, some additional personal information is needed to run the approximation. Note that so far, all of these steps have required [**user input**](https://docs.camunda.io/docs/components/modeler/bpmn/user-tasks/). Next, we make use of a service task which allows you to provide your own code. This is contained in our .NET Camunda worker which is written, for example, in C#.  After completing this step, we then show the user what and how much they need to drink to reach the Ballmer Peak. Then, the process ends.

Following the bottom path, we only run through user tasks. The person chooses and drinks an alcoholic beverage, tests themselves, and in case the Ballmer Peak is not yet reached, they will do exactly these steps over and over again. When the person reaches a BAC above 0.129%, the process ends.

Before we finish up the design phase, we need to design some user interfaces for our user tasks to make the lives of our users a little easier. At Camunda, we therefore design [**forms**](https://docs.camunda.io/docs/components/modeler/forms/camunda-forms-reference/). These can be created using a drag and drop editor. For a developer who avoids doing anything front-end related (like me) this is very helpful.

Figure 3 contains the example form which is appended to the “Enter personal information” user task of our process model. It contains a number input field, which cannot go lower than 40, a default value of 60, and a radio button for gender selection. Both of these elements are required to be filled out by a user. Unfortunately, I could not find an algorithm featuring BAC calculations for nonbinary individuals–please let me know if you are aware of any!

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung*Figure 3: A look at Camunda Forms*

After creating these forms and attaching them to the corresponding user tasks in our process diagram, we are ready to continue implementing our Camunda Platform 8 client. The client encapsulates the worker we’ve previously mentioned and provides even more functionality that your code can leverage to connect to Camunda.

**Implementing Camunda 8 Client with .NET**

In this section we will build a Camunda Platform 8 client with .NET Standard 6.0. Our client should ideally cover the functionality of deploying process definitions, starting process instances, and working on tasks (e.g. for calculating the necessary alcoholic beverages for the Ballmer Peak). The communication between the **[Zeebe](https://camunda.com/platform/zeebe/)** process engine and our application happens via **[gRPC](https://docs.camunda.io/docs/apis-clients/grpc/)**.

Luckily, we don’t need to invest a lot of time and resources to implement the gRPC interface as [**Christopher Kujawa**](mailto:christopher.zell@camunda.com), a senior software engineer at Camunda, already created a library for exactly these purposes. **[Zb-Client](https://github.com/camunda-community-hub/zeebe-client-csharp)** is available on [**NuGet**](https://www.nuget.org/packages/zb-client/) and uses .NET Standard 2.0. To run it, make sure you have either the same or higher version of .NET Standard, .NET Core 2.1+, or .NET Framework 4.7.1+ installed.

Before we can use the client, we have to initialize it. To achieve this, we first need to define and initialize our Zeebe Client. By using the CamundaCloudClientBuilder which comes with the dependency, we can build the zeebeClient. To connect to Camunda Platform 8 we need to provide an ID, a secret, and the contact point. This information can be derived from [**the cluster**](https://docs.camunda.io/docs/guides/orchestrate-microservices/#create-credentials-for-your-zeebe-client) created in the SaaS version.

|  |
| --- |
|  |

**Process deployment**

Now that we have a Zeebe client, let’s take a look at what the actual deployment of the process model to the workflow engine would look like. I have encapsulated that logic inside a separate method which is provided with the location of the BPMN file as a parameter.

By running the NewDeployCommand and handing over the resource file, we are able to handle the deployment. Camunda Platform 8 will return an answer which contains the bpmnProcessId. This one is interesting for us since we want to reuse it to start a new process instance as a next step.

|  |
| --- |
|  |

**Starting a process instance**

After having successfully deployed the process to Zeebe we can now start an instance of it. Once again, I have chosen to create a separate method for the task. The bpmnProcessId returned by the previous method is now used to find the right process. Besides, we are using the NewCreateProcessInstanceCommand which will return the processInstanceKey.

|  |
| --- |
|  |

If we wanted to, we could also have added a payload (e.g. some variables) to the start of the process instance.

**Getting some work done**

After deploying our process diagram and starting an instance, we can implement the worker to take care of approximating the alcohol needed to reach the Ballmer Peak. To do so, we need to add the new worker to our main method.

When initializing a new worker for our zeebeClient, we need to provide the \_JobType which correlates to the type featured in the [**service task**](https://docs.camunda.io/docs/components/modeler/bpmn/service-tasks/) of our BPMN diagram. Additionally, we can specify the timeout of 10 seconds, poll interval of one second, and number of jobs (five) we fetch from Zeebe. Once a task has been retrieved, we can handle it in the method TriggerApproximation described below:

|  |
| --- |
|  |

The business logic is encapsulated in another method which receives the jobClient with access to all job-related operations as well as the job object.

Inside the method, we fetch two JSON objects for gender and weight of a person from the process instance and parse it to basic data types. Afterwards, we can estimate the grams of alcohol needed to reach the Ballmer Peak. Additionally, we run a suggestion algorithm on what kind of beverages to drink to consume that many grams of alcohol.

Next, we complete the job by running the NewCompleteJobCommand on our jobCient. The dictionary containing the suggested drinks is parsed into a JSON object and added to the command. By doing so, this variable is added to the process instance:

Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

|  |
| --- |
|  |

Copy

As demonstrated in this blog post, integrating with Camunda Platform 8 can be rather simple. Typically, you need one access layer and reference to your business logic contained within other projects of the solution. The code we have seen above is often referred to as “[**glue code**](https://en.wikipedia.org/wiki/Glue_code)” as it bridges the gap between the process engine and execution/business logic.

Some parts of what we have seen in this code, like the deployment and start of processes and instances can be also achieved within Web Modeler; this is helpful when collaborating with others or quickly prototyping a solution.

After implementing our Zeebe client, we can take it for a test drive. If you are new to Camunda 8, follow the guide here to [**execute your process diagram**](https://docs.camunda.io/docs/guides/automating-a-process-using-bpmn/#execute-your-process-diagram). Once you’ve executed it, let’s take a look at some impressions of Camunda Platform 8 SaaS.

**Camunda Platform 8 Walkthrough**

Taking our newly built process application on a test run was a success. It is up and running and capable of deploying processes, starting instances, and working on tasks.

[**Tasklist**](https://docs.camunda.io/docs/components/tasklist/introduction-to-tasklist/) is a tool where users can interact with their tasks. In Figure 4 we can see one of the forms created. This form in particular shows the available data to the user.

So to conclude, if you are a male and weigh 75kg, you can drink 65.79 grams of alcohol before reaching the Ballmer Peak. Additionally, the equivalent amount of beer, gin, and wine is displayed, assuming you do not want to drink pure alcohol.

After claiming the task, you can start working on it. This avoids the possibility of two people working on the same task. Once the user drinks the output value, the task can be completed.

Ein Bild, das Text, Screenshot, Zahl, Schrift enthält.

Automatisch generierte Beschreibung*Figure 4: Camunda Platform 8 Tasklist view*

The Operate dashboard for our Ballmer Peak Process visualizes where the created process instances are currently at (figure 5). This tool also has the capability to visualize incidents which might occur and let you control the lifecycle of process instances. Incidents are unforeseen technical problems which the application cannot recover from. For instance we will be able to see an exception thrown in our code.

Ein Bild, das Text, Diagramm, Reihe, Zahl enthält.

Automatisch generierte Beschreibung*Figure 5: Running instances in Operate*

Furthermore, we can also take a look at process instances which have already been completed. This gives us a clue about what happened throughout the execution and what data was relevant for the instance; this is illustrated in Figure 6.

Ein Bild, das Text, Diagramm, Screenshot, Reihe enthält.

Automatisch generierte Beschreibung*Figure 6: Completed instance view in Operate*

Last but not least, let’s take a look at [**Optimize**](https://docs.camunda.io/optimize/components/what-is-optimize/). This tool is used to analyze and improve our running processes. In Figure 7, I have created a dashboard containing some reports to analyze our processes. For instance, how many beers are required on average to reach the Ballmer Peak. I’ve also created some heat maps demonstrating how often each step was run and how long it took. Regarding how long each step took, we can see a spike when it comes to time the approximation took. This was due to the written .NET client being offline. It seemed like the responsible developer in our case hit rock bottom after overpassing the Ballmer Peak. 😉

Ein Bild, das Text, Screenshot, Diagramm, Design enthält.

Automatisch generierte Beschreibung*Figure 7: Analytics in Optimize*

**Conclusion**

Camunda Platform 8 allows you to create a process automation project in just a few steps. It is a [**polyglot solution**](https://docs.camunda.io/docs/apis-clients/working-with-apis-clients/) and works with nearly any language and technology, [**including .NET**](https://docs.camunda.io/docs/apis-clients/community-clients/c-sharp/). By leveraging Operate and Optimize, you are able to gain some visibility which is often key to microservice architectures and processes in general. This is especially interesting for .NET users, who have traditionally been more limited in their options.

Now it is your turn to try out Camunda Platform 8! Do not hesitate to try out the [**SaaS offering for free**](https://signup.camunda.com/accounts). You can clone my project from [**GitHub**](https://github.com/Hafflgav/camunda-8-dotnet-guide), update the [**API credentials**](https://docs.camunda.io/docs/guides/orchestrate-microservices/#create-credentials-for-your-zeebe-client), and discover it by yourselves without writing any code.