

# Upravljanje poslovnim procesima (UPP)

**Nositelj:** izv. prof. dr. sc. Darko Etinger

**Asistent:** Luka Blašković, mag. inf.

**Ustanova:** Sveučilište Jurja Dobrile u Puli, Fakultet informatike u Puli



Fakultet informatike u Puli

## (5) Uvod u procesne aplikacije

#5

UPP

Procesne aplikacije omogućuju automatizaciju poslovnih procesa korištenjem definiranih modela i pravila. Na ovom kolegiju naučili ste kako definirati poslovne procese korištenjem BPMN notacije, a sada ćete se upoznati s alatom Camunda 7 koji omogućuje izvođenje (egzekuciju) tih procesa. Camunda 7 je open-source platforma koja koristi BPMN za vizualno modeliranje procesa te pruža mehanizme za njihovo izvršavanje, nadzor i upravljanje. Primjenom Camunda, organizacije mogu optimizirati svoje poslovne procese i povećati učinkovitost poslovanja kroz automatizaciju zadataka i transparentno praćenje tijeka procesa.

Posljednje ažurirano: 17.12.2024.

## Sadržaj

- [Upravljanje poslovnim procesima \(UPP\)](#)
- [\(5\) Uvod u procesne aplikacije](#)
- [Sadržaj](#)
- [1. Uvod u procesne aplikacije](#)
- [2. Camunda 7](#)
  - [2.1 Pokretanje preko Dockera](#)
- [3. Osnovne komponente Camunda 7 platforme](#)
  - [3.1 Camunda Cockpit](#)
    - [3.1.1 Egzekucija vlastitog procesa](#)
    - [3.1.2 user task i forme](#)
  - [3.2 Camunda Tasklist](#)
    - [3.2.1 Procesne varijable i dodavanje XOR skretnice](#)

- [3.2.2 Kako još možemo izraditi instance?](#)
- [4. Obrada vrijednosti procesnih varijabli u procesu](#)
- [Samostalni zadatak za Vježbu 5](#)

# 1. Uvod u procesne aplikacije

Od početka razvoja BPMN-a isticalo se ostvarenje dvaju (prividno) međusobno teško uskladivih ciljeva:

1. **osigurati** da se normom služe poslovni stručnjaci koji ne razvijaju aplikacije i
2. **omogućiti** softverskim inženjerima da procesni model, izведен po toj normi, preslikaju u izvršnu aplikaciju primjerenu potrebama stvarnoga poslovnog procesa.

Drugim riječima, važna namjena BPMN-a jest premošćivanje jaza u sporazumijevanju između poslovnih i informatičkih stručnjaka 😊

Too often tension exists between the developer and analyst perspectives, resulting from the lack of a common semantics and heuristics set capable of depicting process activities in a way relevant to both parties.

Promatramo li BPMN 2.0 normu općenito, s odmakom od formalno izrečenih logičkih i tehničkih pojedinosti, možemo zaključiti da ona ima sljedeća svojstva:

- sadržava skup **pravila i simbola** za modeliranje poslovnih procesa i omogućuje različite oblike za grafičko predstavljanje procesa, primjereno namjeni
- detaljno razrađen **grafički model** poslovnog procesa može se pretvoriti u izvršiv oblik i na temelju toga razviti potrebna softverska rješenja.
- pogodan za **zajednički jezik za sporazumijevanje** između poslovnih stručnjaka, **analitičara procesa i softverskih inženjera**.

**Procesna aplikacija (PA)** (*eng. process application*) se temelji na tijeku rada, odnosno može se reći da je svaka PA procesno usmjerena (*eng. workflow-oriented*). To je najšira definicija procesne aplikacije. Za preciziranje te definicije prikladno je reći što PA nije, odnosno po čemu se razlikuje od ostalih, podatkovno usmjerениh aplikacija:

**Klasične aplikacije (ne PA)** imaju sljedeća tipična svojstva:

- Funkcionalnosti, koje se ukratko mogu opisati izrekom "**upiši u bazu, pročitaj iz baze**", definirane su stanjem podataka nakon što su izvedene određene aktivnosti ili proveden cijeli proces
- **Redoslijed izvođenja aktivnosti (ili tok rada) implicitno je sadržan u aplikaciji**, obično određen programiranim redoslijedom prikaza korisničkih poziva, odnosno poziva programske sučelja
- **Aktivnosti i procesi ne postoje kao aplikacijski entiteti**
- **Arhitektura klasične aplikacije prilagođena je funkcionalnim CRUD** (*eng. Create, Read, Update, Delete*) potrebama odnosno stvaranju, čitanju, ažuriranju i brisanju podatkovnih zapisa
- Pri izmjeni poslovnih procesa (npr. zbog promjene zakonske regulative), klasične aplikacije treba **temeljito reprogramirati**, napose komponente njihove poslovne logike (*eng. business logic layer*)

S druge strane, **procesne aplikacije (PA)** imaju sljedeća tipična svojstva:

- **Funkcionalnosti su definirane tijekom rada koji aplikacija mora podržavati. Ishodište je za razvoj procesne aplikacije model procesa, dopunjen tako da bude u grupi izvršivih modela**
- Tijekovi rada u aplikaciji eksplicitni su i **neovisni o korisničkim i programskim sučeljima**

- **Aktivnosti i procesi određeni su kao aplikacijski entiteti** čijim se stanjima i definicijom izravno upravlja
- Arhitektura procesne aplikacije prilagođena je reagiranjem na **događaje** (eng. event-driven) i poruke (eng. message-driven) te **upravljanjem tijekom rada** (eng. workflow management)
- Procesne aplikacije mogu sadržavati jednako **korisničke aktivnosti** (eng. user tasks) i **automatizirane aktivnosti** (eng. service tasks)
- Procesne aplikacije **podržavaju više organizacijskih jedinica u organizaciji** i povezuju ih u cjelovit proces koji kupcu ili korisniku daje traženi proizvod ili uslugu
- **Procesne su aplikacije prilagodljive promjenama poslovnih procesa** jer nakon takvih izmjena ne treba reprogramirati aplikacijske komponente, nego procesnu aplikaciju samo opskrbiti ažuriranom definicijom aktivnosti i/ili procesa.

Razlike između klasičnih i procesnih aplikacija mogu se sažeti u sljedećoj tablici:

Svojstva aplikacije	Klasična aplikacija (OLTP ili ERP)	Procesna aplikacija
Funkcionalnosti	Definirane stanjem podataka na kraju posla	Definirane stanjem eksplicitno navedenih radnih aktivnosti. Ishodište za razvoj PA je <b>izvršivi model procesa</b>
Funkcionalna sintagma	"Upiši u bazu, pročitaj iz baze"	"Slijedni najbolji radni tok."
Aktivnosti i procesi	Ne postoji kao programske entitete	<b>Postoje kao programski entiteti</b> kojima se izravno upravlja
Arhitektura	Prilagođena CRUD operacijama	Prilagođena <b>reagiranjima na događaje i poruke</b> (event-driven & message driven)

**Napomena:** U praksi, granica između funkcionalnosti klasične i procesne aplikacije nije uvijek crno-bijela.

#### Primjer ove diferencijacije na webshop aplikaciji:

- **Klasična aplikacija** (zamišljamo u kontekstu kolegija *Programsko Inženjerstvo* ili *Web aplikacije*):
  - Funkcionalnosti implementiramo *low-level* programiranjem gdje razmišljamo o CRUD operacijama nad bazom podataka
  - *Primjer 1:* "Korisnik se registrira i pregledava proizvode" → CRUD operacije nad tablicama `users` i `products`, razvoj korisničkog sučelja, razvoj korespondirajućeg backenda za validaciju podataka i sl.
  - *Primjer 2:* "Korisnik dodaje proizvode u košaricu i obavlja kupnju" → CRUD operacije nad tablicama `cart` i `orders`, razvoj korisničkog sučelja, razvoj odgovarajućih backend komponenti, spajanje na vanjske servise za plaćanje i sl.
  - **Ključno: Aplikacija se organizira oko podataka i operacijama nad njima**

- **Procesna aplikacija** (zamišljamo u kontekstu ovog kolegija):
  - Funkcionalnosti implementiramo *high-level* programiranjem gdje razmišljamo o **tijeku rada i aktivnostima koje korisnik treba obaviti**
  - *Primjer 1:* "Korisnik se registrira i pregledava proizvode" → Procesna aplikacija definira radne korake koje korisnik treba obaviti, npr. "Registracija korisnika", "Pregled proizvoda", "Dodavanje proizvoda u košaricu". Korake definiramo kroz neki **procesni model** (u našem slučaju **BPMN**, ali može biti i drugi)
  - **Aplikacija reagira na vanjske događaje i poruke** (npr. "Korisnik je pokrenuo proces narudžbe", "Plaćanje uspješno", "Proizvod je otpremljen") → *event-driven* i *message-driven* programiranje
  - Proces narudžbe zamišljamo kao **jedan entitet** koji se sastoji od više koraka (aktivnosti) koje korisnik treba obaviti, gdje određeni koraci mogu biti različitih tipova (korisničke aktivnosti, automatizirane aktivnosti, itd.).
  - **Ključno: Jedno započinjanje aktivnosti nazivamo pokretanje procesne instance.** Npr. "korisnik Pero Perić započinje proces narudžbe u webshopu". Samim time, svaki korisnik procesne aplikacije koji započne proces ima svoju **instancu procesa**. Stanje instance procesa može se ponovno izgraditi u svakom trenutku koristeći tzv. *event logove*.

Ipak, svim komercijalnim alatima za razvoj procesnih aplikacija, kao što je Camunda, zajedničko je sljedeće svojstvo: **procesne aplikacije izvode se kao web aplikacije**.

U kontekstu ovog kolegija, procesne aplikacije ćemo interpretirati kao web aplikacije koje koriste Camunda 7 platformu za izvođenje poslovnih procesa definiranih BPMN 2.0 normom.

Do sad smo naučili kako ispravno modelirati procese u BPMN notaciji, a sada ćemo se upoznati s alatom Camunda 7 koji omogućuje izvođenje (egzekuciju) tih procesa.

## 2. Camunda 7

Camunda 7 je *open-source* platforma koja koristi BPMN za vizualno modeliranje procesa te pruža mehanizme za njihovo **izvršavanje, nadzor i upravljanje**. Primjenom Camunde, organizacije mogu optimizirati svoje poslovne procese i povećati učinkovitost poslovanja kroz automatizaciju zadataka i transparentno praćenje tijeka procesa.



Do sad ste koristili [Open Source Desktop Modeler](#) za modeliranje poslovnih procesa u BPMN notaciji, sada ćemo se upoznati s dodatnim komponentama Camunda platforme:

- **BPMN Workflow Engine**
- **DMN Decision Engine**
- **Tasklist**
- **Cockpit i Admin**

Camunda 7 nudi besplatni Community Edition paket koji uključuje sve potrebne komponente za upoznavanje procesnih aplikacija i njihovu egzekuciju lokalno na računalu, dok Enterprise Edition paket nudi dodatne mogućnosti za upravljanje i nadzor poslovnih procesa u velikim organizacijama, optimizacijske tehnike i naprednije sigurnosne mehanizme.

Također treba naglasiti da je Camunda 7 platforma EOL (End of Life) te je zadnje ažuriranje dobila u 10. mjesecu 2024. godine. Međutim, radi se o dobro razrađenom softveru koji se i dalje može koristiti za učenje i razvoj procesnih aplikacija. Camunda aktivno radi na razvoju nove verzije Camunda 8 koja donosi brojne nove mogućnosti.

Radi jednostavnije instalacije, ali i **dostupnosti materijala za učenje**, preporučuje se korištenje **Camunda 7** platforme do daljnega.

### 2.1 Pokretanje preko Dockera

Camunda 7 platformu možete vrlo jednostavno pokrenuti lokalno preko [gotovog Docker kontejnera](#).

**Docker** je besplatna platforma koja omogućuje razvoj, postavljanje i pokretanje aplikacija u kontejnerima (eng. [Containerization](<https://en.wikipedia.org/wiki/Containerization\_(computing)>)). Kontejneri omogućuju pakiranje i izvršavanje aplikacija u izoliranom okruženju, što znači da se aplikacija može pokrenuti na bilo kojem računalu koje ima Docker instaliran, bez obzira na okruženje.

Prvi korak je preuzimanje **Docker Desktop** aplikacije sa [službene stranice](#).

Ako ste na Windows OS-u, Docker Desktop zahtjeva instalaciju WSL-2 (Windows Subsystem for Linux) koji se može instalirati preko PowerShell naredbe:

```
wsl --install
```

Dodatno, moguće je omogućiti **Hyper-V i podršku za virtualizaciju** u BIOS-u računala. Ovisno o proizvođaču matične ploče, postupak se razlikuje, ali BIOS-u se obično pristupa pritiskom tipke **F2**, **F10**, **F12** ili **DEL** prilikom pokretanja računala.

Najbolji način je pretražiti na internetu kako pristupiti BIOS-u na vašem računalu te kako omogućiti Hyper-V i virtualizaciju.

Dakle, na Windowsu, Docker Desktop zahtjeva instalaciju WSL-2 **ili** Hyper-V.

## System requirements



### Should I use Hyper-V or WSL?

Docker Desktop's functionality remains consistent on both WSL and Hyper-V, without a preference for either architecture. Hyper-V and WSL have their own advantages and disadvantages, depending on your specific set up and your planned use case.

[WSL 2 backend, x86\\_64](#)   [Hyper-V backend, x86\\_64](#)   [WSL 2 backend, Arm \(Beta\)](#)

- WSL version 1.1.3.0 or later.
- Windows 11 64-bit: Home or Pro version 22H2 or higher, or Enterprise or Education version 22H2 or higher.
- Windows 10 64-bit: Minimum required is Home or Pro 22H2 (build 19045) or higher, or Enterprise or Education 22H2 (build 19045) or higher.
- Turn on the WSL 2 feature on Windows. For detailed instructions, refer to the [Microsoft documentation](#).
- The following hardware prerequisites are required to successfully run WSL 2 on Windows 10 or Windows 11:
  - 64-bit processor with [Second Level Address Translation \(SLAT\)](#)
  - 4GB system RAM
  - Enable hardware virtualization in BIOS. For more information, see [Virtualization](#).

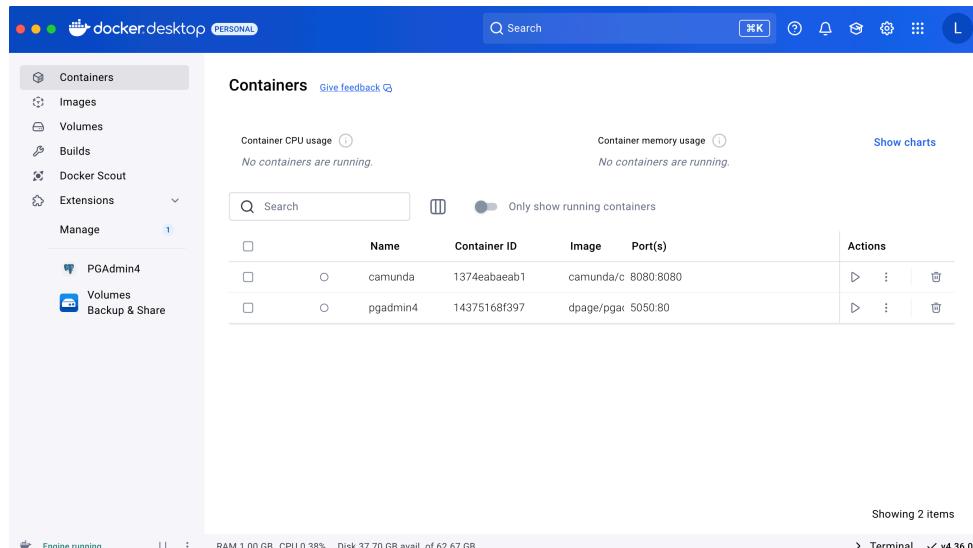
Upute za instalaciju Docker Desktop na Windows OS-u, dostupno na: <https://docs.docker.com/desktop/setup/install/windows-install/>

Docker je moguće koristiti i na **Linux** (dostupno za: Ubuntu, Debian, RHEL, Fedora) i **macOS** (dostupno za: Apple silicon, Intel chip) operacijskim sustavima bez dodatnih postavki. [Na Linuxu možete instalirati Docker i bez grafičkog sučelja preko terminala](#), međutim za početnike je preporuka instalirati grafičko sučelje - **Docker Desktop**.

Nakon što ste uspješno instalirati **Docker Desktop**, provjerite je li uspješno instaliran preko naredbe:

```
docker --version
```

Pokrenite Docker Desktop aplikaciju i prijavite se s vašim Docker računom. Ako nemate Docker račun, možete ga besplatno kreirati na [Docker Hub-u](#).



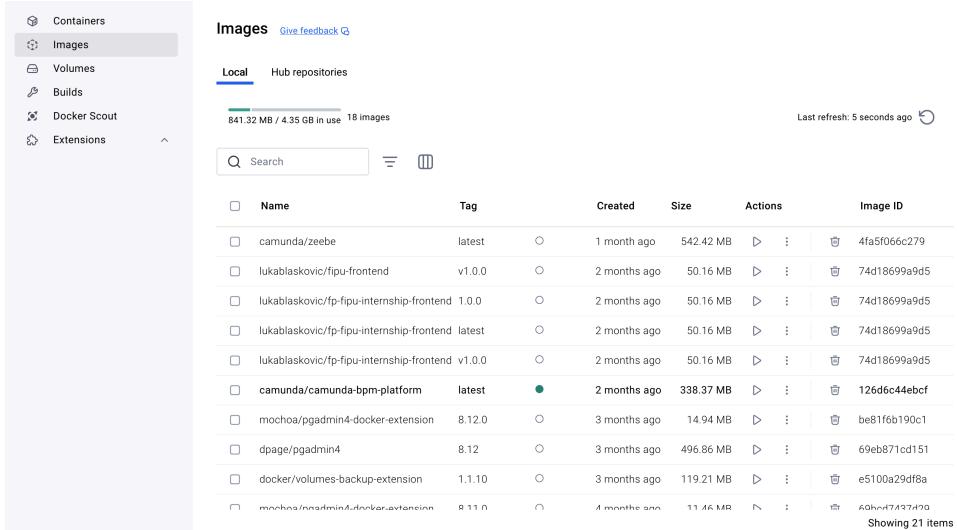
### Grafičko sučelje Docker Desktop aplikacije

Grafičko sučelje Docker Desktop aplikacije sastoji se od nekoliko osnovnih elemenata:

1. **Container** - prikaz svih pokrenutih kontejnera (dva stanja: **Running** i **Exited**). **Kontejner** smo rekli da je svaka aplikacija koja se pokreće u izoliranom okruženju. U ovom slučaju, to će biti Camunda 7 platforma.
2. **Images** - prikaz svih preuzetih Docker "slika" (eng. *Docker images*). **Docker image je predložak za pokretanje kontejnera**.
3. **Volumes** - prikaz svih Docker "volumena" (eng. *Docker volumes*). **Docker volume koristi se za trajno pohranjenje podataka, obzirom da se podaci unutar kontejnera brišu prilikom gašenja kontejnera**.
4. **Builds** - prikaz svih provedenih Docker "buildova" (eng. *Docker builds*). Ovdje su pohranjeni svi Docker buildovi, uspješni i neuspješni.
5. **Docker Scout** - napredna analiza docker images, u svrhu pronalaska ranjivosti (eng. *vulnerabilities*). Za početnike, ovo nije bitno.
6. **Extensions** - dodatne ekstenzije za Docker Desktop aplikaciju. Za početnike, ovo nije bitno.

U pravilu, za sada će nam samo biti bitni **Container** i **Images** tabovi.

**VAŽNO!** Kontejneri se uvijek pokreću preko odgovarajućeg image-a, gdje image predstavlja predložak za pokretanje kontejnera, a kontejner predstavlja realiziranu sliku (instancu) tog predloška.



Prikaz svih preuzetih Docker "slika" (eng. *Docker images*)

Docker Desktop aplikacija prikazuje sve definirane Docker images koje izradimo **(1) lokalno putem terminala ili (2) koje preuzimamo s [Docker Hub-a](#).**

Docker images lokalno gradimo koristeći **Dockerfile** datoteku, koja definira korake kontejnerizacije naše aplikacije. Ovime ćemo se detaljno baviti na kolegiju Raspolijeljeni sustavi na 1. godini diplomskog studija.

Za sada nas zanima samo dio koji se odnosi na preuzimanje gotove Docker "slike" s Docker Hub-a. **Docker Hub** je centralno mjesto za pronaštač i dijeljenje Docker images, slično kao što je GitHub centralno mjesto za pronaštač i dijeljenje izvornog koda (u obliku repozitorija).

Sliku s Docker Hub-a možemo preuzeti preko naredbe u terminalu (možete bilo gdje otvoriti terminal)

```
docker pull <image-name>:<tag>
```

Preuzet ćemo sljedeći [Camunda 7 image](#):

```
docker pull camunda/camunda-bpm-platform:latest
```

Nakon što preuzmemos sliku, možemo je pokrenuti preko naredbe:

```
docker run -d --name camunda -p 8080:8080 camunda/camunda-bpm-platform:latest
```

- `-d` - pokreće kontejner u pozadini (eng. *detached mode*)
- `--name camunda` - dodjeljuje naziv kontejneru
- `-p 8080:8080` - mapira port `8080` na lokalnom računalu (vašem) na port `8080` unutar kontejnera
- `camunda/camunda-bpm-platform:latest` - image koji pokrećemo

Nakon što pokrenemo kontejner, možemo provjeriti je li kontejner uspješno pokrenut preko Docker Desktop aplikacije.

Kontejner je uspješno pokrenut i radi na portu 8080

Kontejner je, osim preko terminala, **moguće pokrenuti direktno iz grafičkog sučelja**, međutim onda je ove dodatne postavke potrebno unijeti u grafičkom sučelju.

### Praktičnije je i brže naučiti osnovne Docker naredbe u terminalu

CLI Cheat Sheet za Docker možete preuzeti na sljedećoj poveznici: [https://docs.docker.com/get-started/docker\\_cheatsheet.pdf](https://docs.docker.com/get-started/docker_cheatsheet.pdf)

Nakon što je kontejner uspješno pokrenut, možemo pristupiti Camunda 7 platformi preko web preglednika na adresi: `http://localhost:8080/camunda`

Otvorite u web pregledniku adresu: `http://localhost:8080/camunda-welcome/index.html`. Ako je kontejner uspješno pokrenut, trebali biste vidjeti sljedeći prikaz:

Camunda 7 platforma pokrenuta lokalno preko Docker kontejnera: `camunda/camunda-bpm-platform:latest`

# 3. Osnovne komponente Camunda 7 platforme

Camunda 7 platforma koju ste pokrenuli sastoji se nekoliko ključnih komponenti:

- **Workflow Engine:** Pozadinski "motor" koji izvršava poslovne procese definirane BPMN 2.0 normom
- **Cockpit:** Monitoring i nadzor poslovnih procesa i aktivnih instanci
- **Tasklist:** Popis zadataka koje korisnici trebaju obaviti (User Tasks)
- **Admin:** Upravljanje korisnicima (User Management), grupama, ovlastima i sl.

Za samu egzekuciju procesa dovoljan je samo **Workflow Engine**, dok su **Cockpit**, **Tasklist** i **Admin** dodatne komponente koje olakšavaju upravljanje i nadzor poslovnih procesa kroz grafička sučelja.

Kao 5. komponentu ne smijemo izostaviti naš **Camunda Modeler** koji koristimo kao odvojeni alat za **modeliranje**. Međutim, vidjet ćete da modeliranje sad dobiva na još jednoj dimenziji - onoj **podatkovnoj**.

Za učenje, ovdje već imate uključena 2 poslovna procesa:

- *Invoice Receipt*
- *Review Invoice*

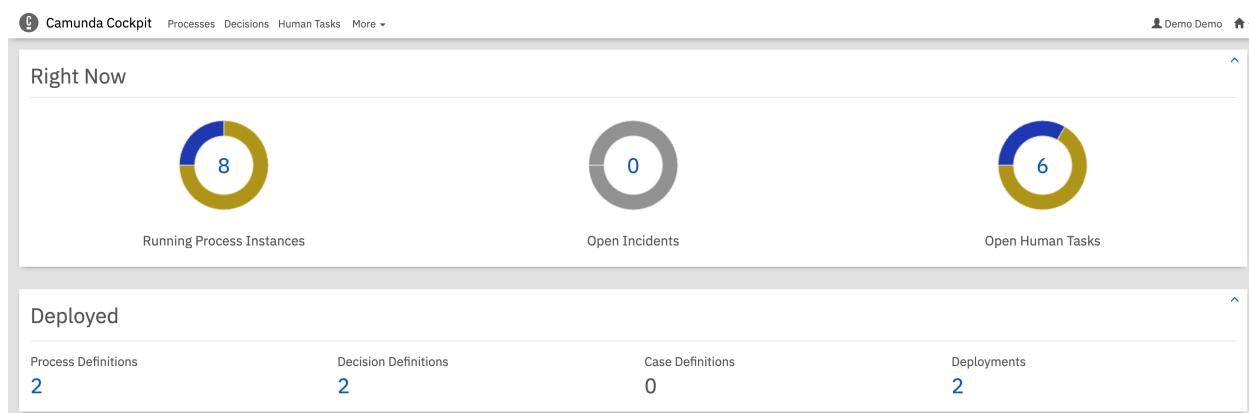
## 3.1 Camunda Cockpit

Na početnoj stranici, gdje vidite prikaz svih komponenti, odaberite **Cockpit**.

Tražit će vas da se prijavite. Korisničko ime i lozinka su `demo`.

**Camunda 7 Cockpit** predstavlja centralno mjesto za **nadzor, upravljanje i analizu poslovnih procesa** koji se izvršavaju. Kao i druge komponente (osim Workflow Enginea), Cockpit je dostupan preko web grafičkog sučelja.

Jednom kad se prijavite, vidjet ćete upravljačku ploču koja prikazuje trenutne **aktivne procese i procesne instance**.



Početna upravljačka ploča Camunda Cockpit komponente

Podsjetimo se: **Procesna instanca odnosi se na jedno pokretanje procesa**.

Svaki korisnik koji pokrene proces ima svoju instancu procesa.

Na ovom prikazu vidimo 2 aktivna procesa:

- *Invoice Receipt*
- *Review Invoice*

Od toga, postoji:

- 6 aktivnih instanci procesa *Invoice Receipt* i
- 2 aktivne instance procesa *Review Invoice*

Ako pritisnute **Processes** u gornjem izborniku, vidjet ćete popis pokrenutih **procesa**.

Camunda Cockpit

Processes Decisions Human Tasks More ▾

2 process definitions deployed

Add criteria	State	Incidents	Running Instances	Key	Name	Tenant ID
	<input checked="" type="checkbox"/>	0	6	invoice	Invoice Receipt	
	<input checked="" type="checkbox"/>	0	2	ReviewInvoice	Review Invoice	

### Pregled aktivnih procesa i procesnih instanci

Otvorite proces *Review Invoice* kako biste vidjeli aktivne instance tog procesa i trenutno stanje procesa kroz **dijagram toka**.

Dashboard » Processes » Review Invoice : Runtime

Definition Version: 1

Version Tag: null

Definition ID: ReviewInvoice:1:37dddefa2-b6fc-11ef...

Definition Key: ReviewInvoice

Definition Name: Review Invoice

History Time To Live: 45 days

Tenant ID: null

Deployment ID: 37a923fc-b6fc-11ef-9b3c-0242ac1...

Instances Running:

- current version: 2
- all versions: 2

Activity Instance Statistics: on

Process Instances Incidents Called Process Definitions Job Definitions

Add criteria	State	ID	Start Time	Business Key
	<input checked="" type="checkbox"/>	393f0622-b6fc-11ef-9b3c-0242ac110002	2024-12-10T14:39:47	
	<input checked="" type="checkbox"/>	3926271a-b6fc-11ef-9b3c-0242ac110002	2024-12-10T14:39:47	

Uočite putanju u programu:

Dashboard → Processes → Review Invoice : Runtime

Sad **pregledavamo aktivne instance procesa** *Review Invoice*. Vidimo da postoje dvije.

U prvom planu vidimo dijagram ovog poslovnog procesa koji se sastoji od samo **2 aktivnosti (eng. Task)**:

- *Assign Reviewer* (User Task)
- *Review Invoice* (User Task)

Na dijagramu vidimo brojku **(2)** pored aktivnosti *Assign Reviewer* što znači da postoje **2 aktivne instance** ovog procesa **koje se trenutno nalazi na ovoj aktivnosti** (radnom koraku).

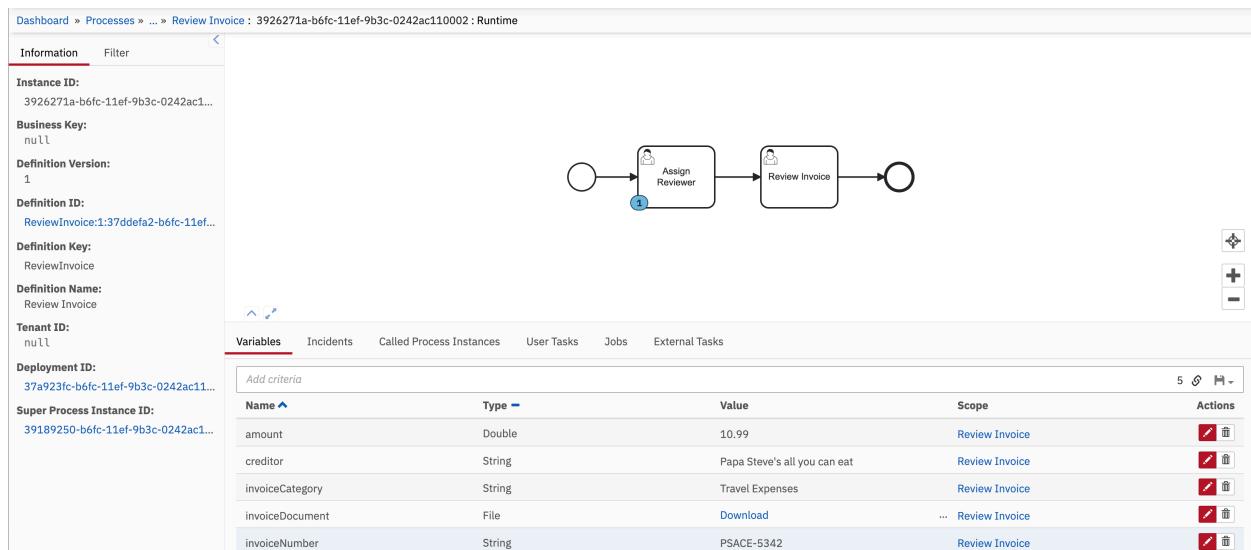
Spomenutu oznaku nazivamo **token**.

U izborniku lijevo možemo vidjeti neke općenite informacije o procesu, kao što su:

- **Definition Version** - verzija definicije procesa (u slučaju da se proces mijenja, što je čest slučaj u praksi)
- **Definition ID** - ID trenutne definicije poslovnog procesa
- **Definition Key** - ključ definicije procesa
- **History Time To Live** - koliko dugo se povijest procesa (pohranjene procesne varijable) čuva u internoj bazi podataka
- **Deployment ID** - ID trenutnog *deploymenta* poslovnog procesa
- **Instance Running** - koliko je trenutno aktivnih instanci procesa, za trenutnu verziju i sve verzije ukupno

Ako pritisnemo na jednu od dvije instance, otvorit će se još jedan prozor s detaljima o toj instanci. Ovdje su nam najzanimljivije pohranjene **procesne varijable**:

- *amount*
- *creditor*
- *invoiceCategory*
- *invoiceDocument*
- *invoiceNumber*



Detalji o jednoj instances procesa *Review Invoice* (**procesne varijable**)

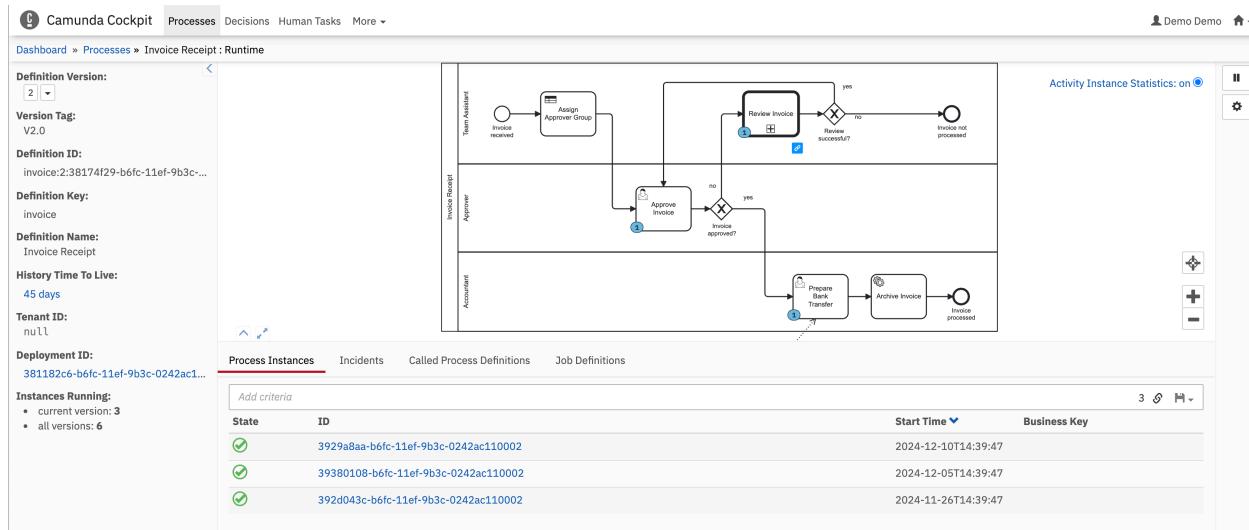
Vidimo da se svaka varijabla sastoji od:

- **naziva/ključa**
- **tipa podataka** (npr. `Integer`, `Boolean`, `Double`, `String`, `File`, itd.)
- **vrijednosti**
- **opseg-a procesa gdje je procesna varijabla vidljiva (scope)**

Vrijednosti se ovdje mogu direktno mijenjati, što je korisno **u slučaju da je potrebno ručno intervenirati u procesu**.

U prozoru `Add criteria` moguće je definirati kriterije za filtriranje podataka po **instanci, ključu i vrijednosti varijable**.

Ako se vratimo na `Dashboard → Processes` i otvorimo drugi proces *Invoice Receipt*, možemo vidjeti i ovaj proces i njegove aktivne instance.



### Pregled aktivnih instanci procesa *Invoice Receipt*

Vidimo da je proces složeniji od prethodnog, sastoji se od 3 staze (swimlanes) koje predstavljaju različite sudionike u procesu:

- **Team Assistant**
- **Approver**
- **Accountant**

Ovaj proces ima ukupno 6 aktivnih instanci, od kojih su:

- 3 u definiciji procesa V1.0 i
- 3 u definiciji procesa V2.0

Dalje, možemo uočiti nekoliko aktivnosti na svakoj stazi:

- *Assign Approver Group*
- *Approve Invoice*
- *Review Invoice*
- *Prepare Bank Transfer*
- *Archive Invoice*

Uočavate li nešto? *Review Invoice* je ustvari **potproces** koji se koristi u ovom procesu, međutim on je definiran i *deployan* kao zasebni proces koji se izvršava u Workflow engineu, a i vidjeli smo ga malo prije.

Dakle, kroz Camunda Cockpit, osim glavnog procesa koji se izvršava, **možemo na jednak način pratiti i potprocese koji se izvršavaju unutar glavnog procesa**.

Ako pogledate ovdje graf, možemo vidjeti ukupno 3 tokena: brojke `(1)` na aktivnostima:

- *Approve Invoice*
- *Prepare Bank Transfer*
- *Review Invoice*

Što to znači? 🤔

---

- **Ukupno 1 instance** procesa *Invoice Receipt* je trenutno na aktivnosti ***Approve Invoice***.
  - Npr. "Za invoice broj 12345, Approver mora odobriti račun".
- **Ukupno 1 instance** procesa *Invoice Receipt* je trenutno na aktivnosti ***Prepare Bank Transfer***.
  - Npr. "Za invoice broj 54321, Accountant mora pripremiti bankovni transfer".
- **Ukupno 1 instance** procesa *Invoice Receipt* je trenutno na aktivnosti (potprocesu) ***Review Invoice***.
  - Npr. "Za invoice broj 67890, Team Assistant mora pregledati račun".

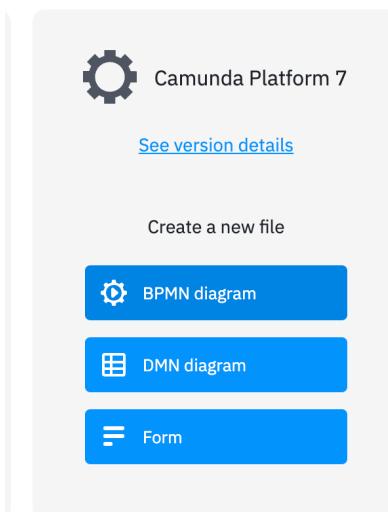
Ako stisnemo na neku od ovih aktivnosti s tokenom, **filtrirat će nam se one instance koje se trenutno nalaze na toj aktivnosti.**

Dodatno, pored potprocesa možemo odabrati opciju `"Show Called Process Definition"` koja će nam otvoriti novi prozor **s detaljima o tom potprocesu i njegovim aktivniminstancama**, dakle klikom na "Review Invoice", otvorit će se `Dashboard -> Processes -> Invoice Receipt -> Review Invoice: Runtime`.

### 3.1.1 Egzekucija vlastitog procesa

Prije nego što krenemo pregledavati druge komponente (`Tasklist`, `Admin`), idemo pokušati definirati vlastiti poslovni proces, pokrenuti ga te pratiti njegovo izvođenje unutar Camunda Cockpita.

Otvorite Camunda Modeler i odaberite novi BPMN dijagram za Camunda 7 platformu.



| Odaberite Camunda Platform 7 -> BPMN diagram

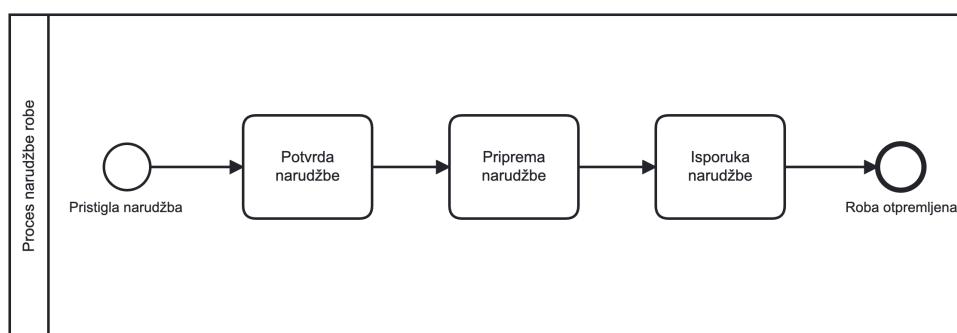
Dijagram pohranite lokalno, na proizvoljnu lokaciju.

Definirat ćemo jednostavan proces koji definira **obradu narudžbe proizvoda u webshopu**. Nazovite ga: `webshop-order.bpmn`

**Krenimo jednostavno**, definirat ćemo proces koji se sastoji od 3 aktivnosti u jednoj stazi:

- *Potvrda narudžbe*
- *Priprema narudžbe*
- *Isporuka narudžbe*

Za sada nećemo definirati dopunske atribute aktivnosti, niti skretnice. Napravite jednostavan linearni proces s 3 aktivnosti i početnim i završnim događajem.



| Jednostavan proces narudžbe u webshopu

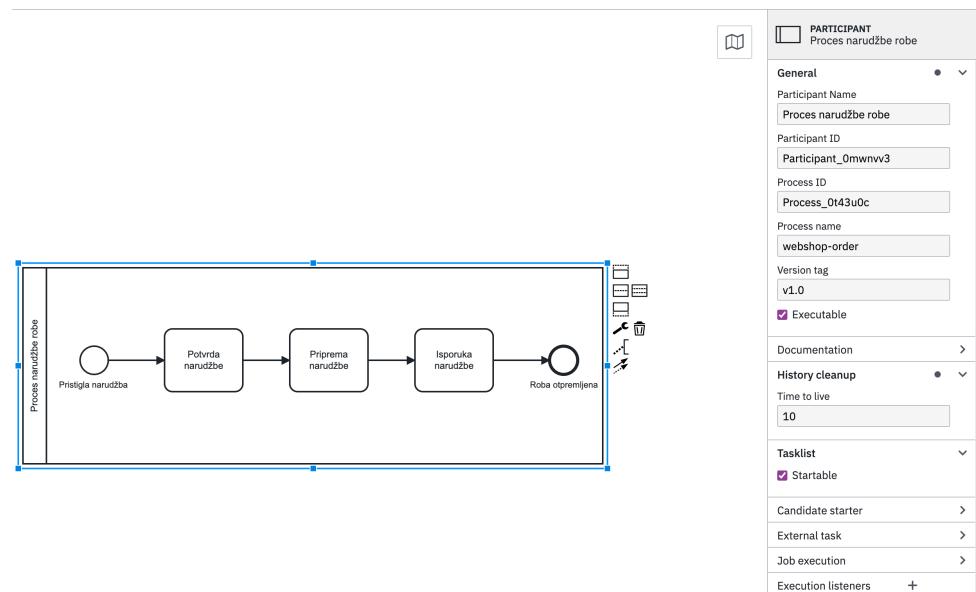
Prije nego možemo *deployati* ovaj proces, moramo definirati nekoliko stvari:

- **Process name** - ime procesa (npr. *Webshop Order*)

- **Version tag** - verzija procesa (npr. v1.0)
- **Time to live** - koliko dugo se povijest procesa čuva u bazi podataka (unesite proizvoljnu vrijednost)
- **Process ID** - jedinstveni identifikator procesa (npr. *narudzba\_robe*) (čisto da vidite da ne mora biti isto kao ime procesa, ovaj podatak ćemo kasnije najviše koristiti)

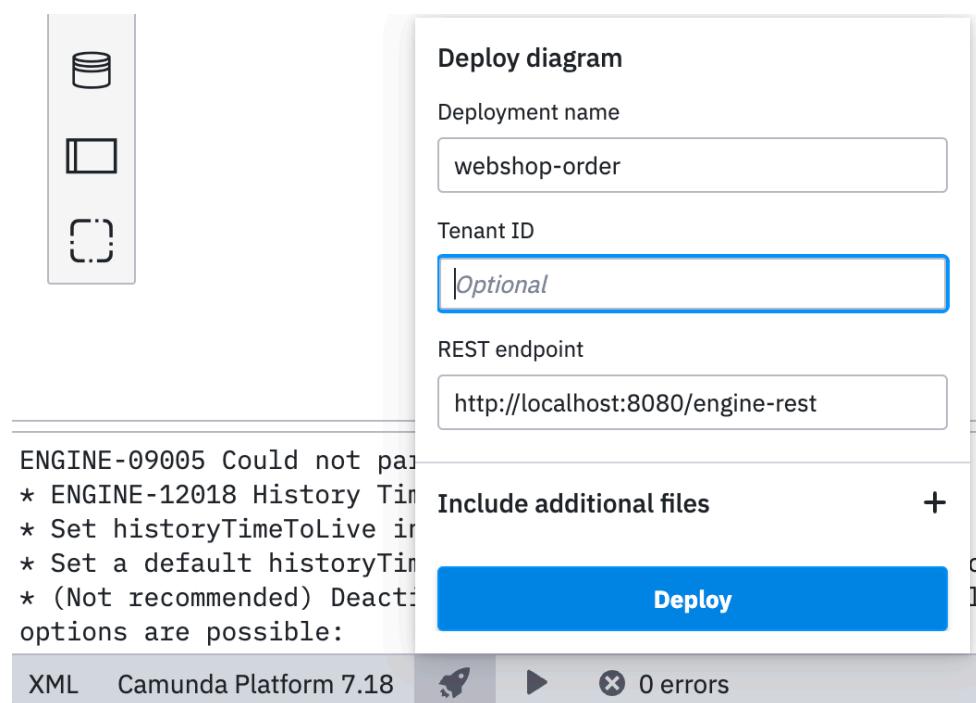
Stisnite na *pool* gdje je sadržan proces, trebao bi vam se otvoriti s desne strane prozor s postavkama procesa (**Properties panel**).

Ako vam se ovaj prozor ne prikazuje, odaberite `Window -> Toggle Properties Panel`. Jednom kad se otvori, **unesite tražene vrijednosti**:



### Postavke procesa u **Properties panelu**

Sada možete deployati diagram pritiskom na ikonu rakete (🚀) u donjem lijevom kutu



Provjerite da se PORT REST endpointa poklapa s portom na kojem je pokrenuta Camunda platforma, odnosno PORT na koji je mapiran Docker kontejner.

Trebali biste dobiti poruku o uspješnom deploymentu definicije procesa. Sada otvorite Camunda Cockpit i pregledajte procese:

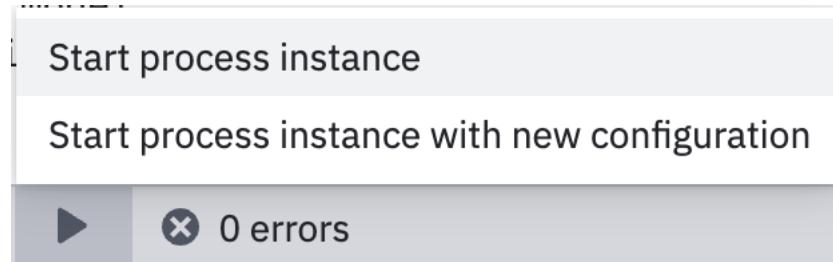
The screenshot shows the Camunda Cockpit interface with the 'Processes' tab selected. The main area displays a table titled '3 process definitions deployed'. The columns are: State, Incidents, Running Instances, Key, Name, and Tenant ID. The data rows are:

State	Incidents	Running Instances	Key	Name	Tenant ID
Green checkmark	0	6	invoice	Invoice Receipt	
Green checkmark	0	2	ReviewInvoice	Review Invoice	
Green checkmark	0	0	Process_0143u0c	webshop-order	webshop-order

Ako otvorite proces, vidjet ćete da **nema aktivnih instanci**. To je zato što nismo pokrenuli niti jednu.

U realnom okruženju, proces će se pokrenuti nekim događajem ili korisničkom interakcijom. Međutim, tijekom razvoja procesne aplikacije, možemo ručno pokrenuti proces:

**Vratite se u Modeler**, na dnu odaberite strelicu pored ikone rakete ( ) i odaberite `Start process instance`.



Pokretanje nove instance procesa direktno unutar **Camunda Modelera**

Trebali biste dobiti obavijest o uspješnom pokretanju instance procesa.

Ako se vratite u Camunda Cockpit i osvježite stranicu očekivali biste novu instancu procesa koja čeka na zadatku `"Potvrda narudžbe"`, međutim to nije slučaj. Zašto?

### 3.1.2 User Task i forme

Nećete vidjeti novu instancu procesa, niti će se ona pojaviti u Cockpitu jer je instanca već završila (odnosno započela, odradila zadatke i završila). Naime, proces koji smo definirali je jednostavan linearni proces koji se sastoji od 3 aktivnosti i završnog događaja. Ove 3 aktivnosti koje smo definirali nemaju nikakve dodatne postavke/atribute, odnosno **nismo definirali način na koji korisnik može dati "input" u proces**, npr. **obraditi narudžbu, pripremiti narudžbu** i sl.

Ako se prisjetite, rekli smo da za radnje koje obavlja korisnik preko informacijskog sustava (u našem slučaju je to Camunda Cockpit) koristimo **korisnički radni korak/zadatak** (`User task`) kao **opisni atribut aktivnosti**.

Izmijenit ćemo proces tako da aktivnost `"Potvrda narudžbe"` bude `User task`.

Međutim to nije sve, moramo definirati i neki način kako će korisnik potvrditi tu narudžbu, kojom interakcijom? To ćemo definirati preko **formi**. Camunda 7 dozvoljava da definirate 3 vrste formi:

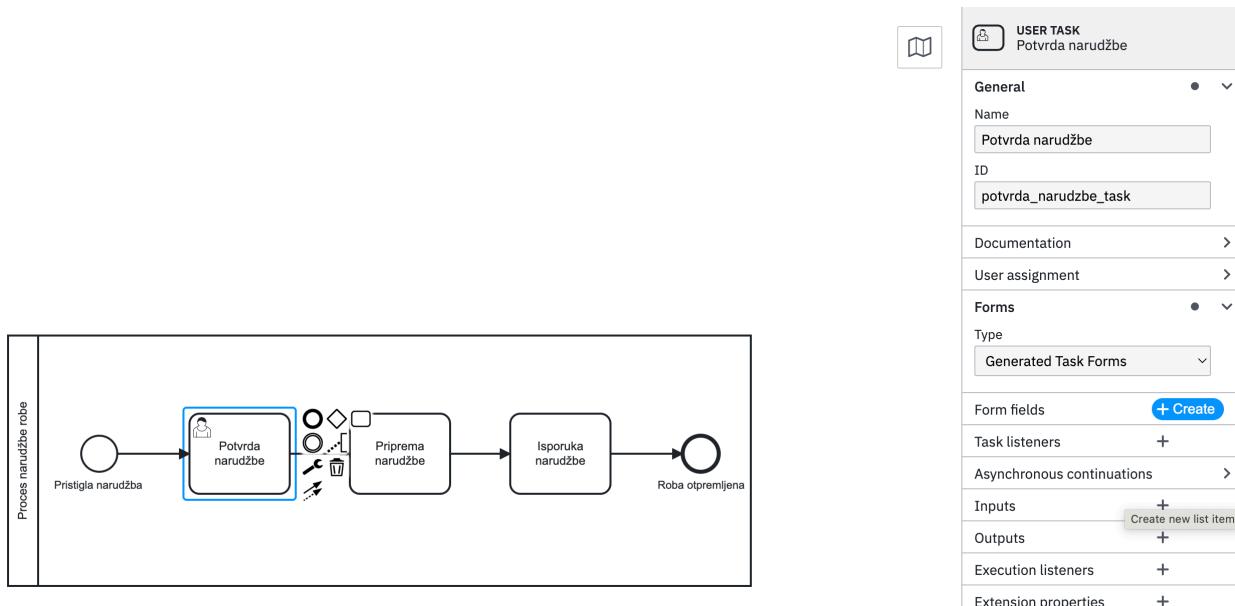
1. `Camunda Forms`
2. `Embedded or External Task Forms`

### 3. Generated Task Forms

Općenito, forme definiraju **način na koji korisnik može unijeti podatke u proces**.

1. Camunda Forms su bazirane na JSON zapisu i direktno su integrirane u Camunda Modeler kroz **Form Editor** (File -> New file -> Form (Camunda platform 7)). Potrebno je dodati referencu na formu, a one će se automatski prikazati u Cockpitu.
2. Embedded or External Task Forms su forme koje se mogu definirati izvan Camunda Modelera, npr. preko HTML/CSS/JS. S njima je moguće komunicirati preko REST API poziva.
3. Generated Task Forms su forme koje se generiraju automatski na temelju varijabli koje su definirane u opcijama **Form fields** u Properties panelu za **User Task**.

Mi ćemo za sad odabrati **treću opciju** obzirom da je ujedno i najjednostavnija.



Unos ID-a i odabir tipa forme: "**Generated Task Forms**"

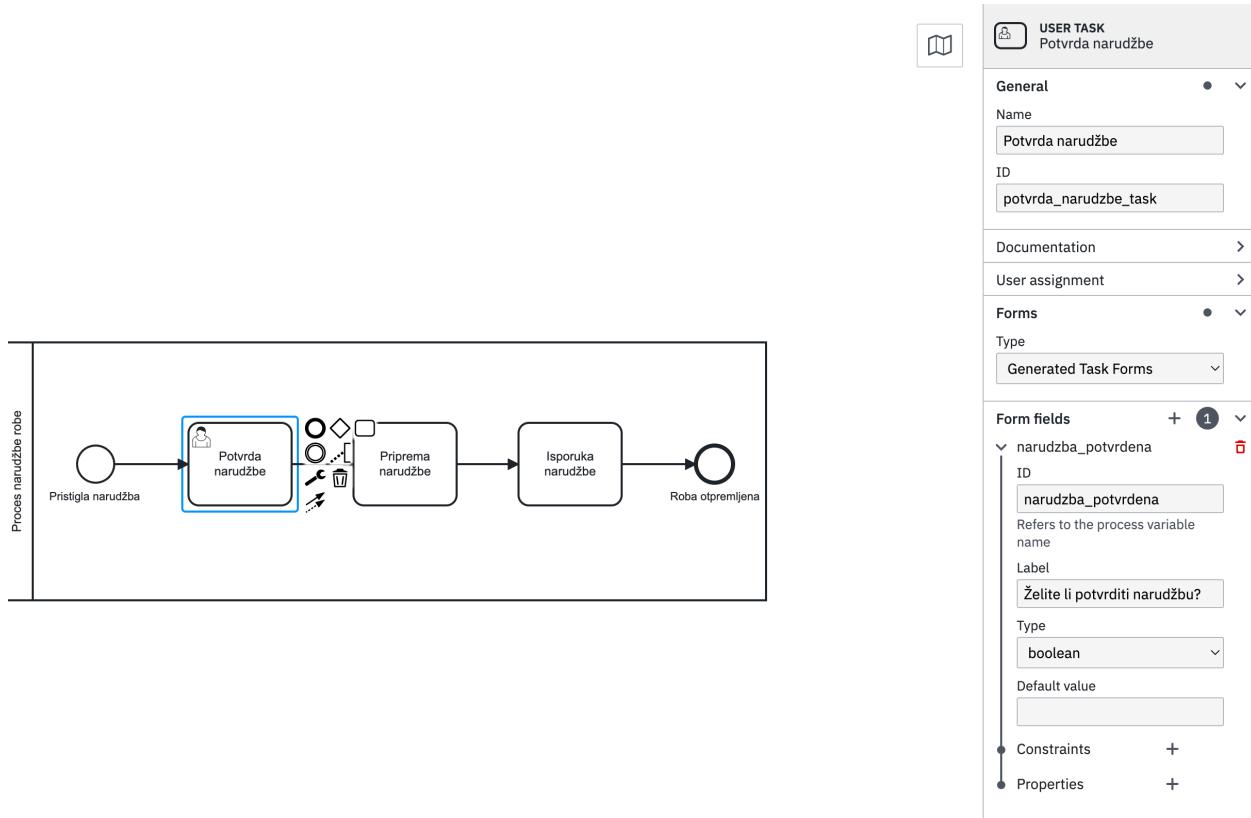
Svakoj aktivnosti, pa i ovoj, možete dodati ID, npr. `potvrda_narudzbe_task`.

1. Pod **Forms** odaberite **Generated Task Forms**
2. Odaberite **Form Fields** i dodajte jedno polje. Nazovite ga `narudzba_potvrdena` i postavite tip podatka na `Boolean`. Pod **Label** možete unijeti labelu koju će korisnik vidjeti za navedeno polje, npr. "Želite li potvrditi narudžbu?".

Uočite dodatne opcije (`Constraints`, `Properties`), za sada ćemo ih ignorirati.

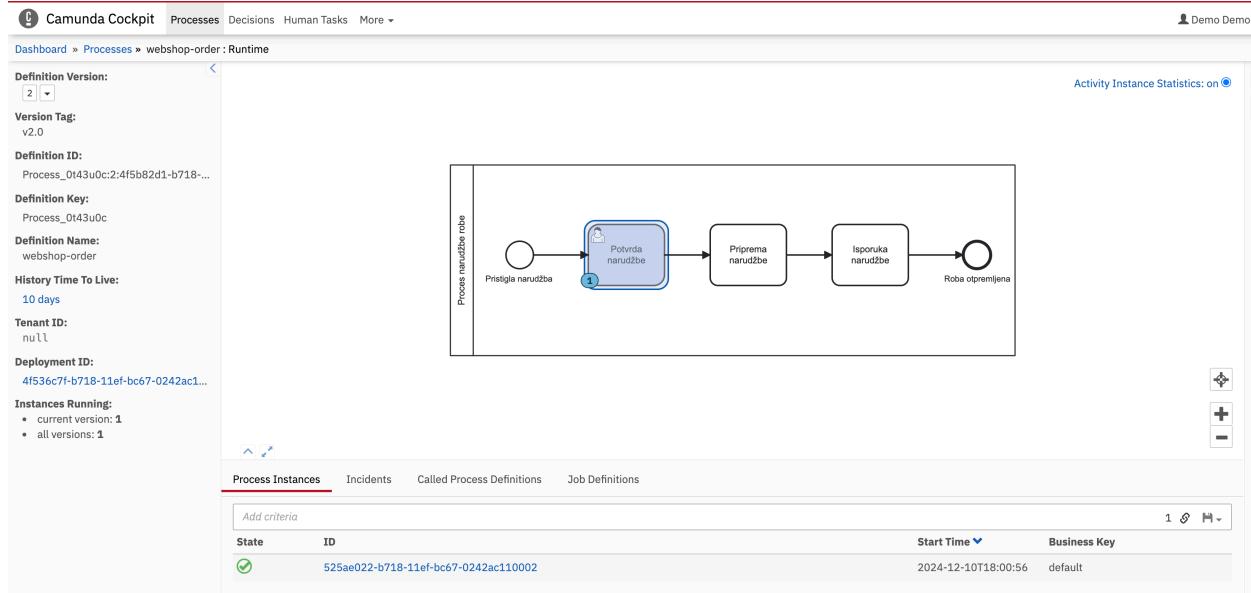
Dignite verziju procesa na `v2.0` i *deployajte* ga.

Nakon toga, startajte novu instancu procesa kroz Modeler.



## Dodavanje novog polja u formu

Otvorite instance ovog procesa, vidjet ćete **1 aktivnu instancu** koja ima token na aktivnosti "Potvrda narudžbe".



Pregled aktivne instance procesa s tokenom na aktivnosti "Potvrda narudžbe"

Kako bi završili ovu aktivnost, moramo potvrditi narudžbu putem nove forme koju smo definirali 😎

## 3.2 Camunda Tasklist

**Camunda Tasklist** je web aplikacija unutar Camunde 7 koja omogućuje korisnicima da pregledavaju i obavljaju korisničke zadatke (**User Task**) koji su im dodijeljeni u poslovnim procesima.

Otvorite **Tasklist** preko početne stranice Camunda platforme.

Ako vas traži korisničke podatke, možete ponovo unijeti **demo** kao korisničko ime i lozinku.

Lijevo ćete vidjeti grupirane zadatke po kategorijama. Zadatke je moguće grupirati po korisnicima, po procesima, po prioritetu, po datumu, itd. Možete vidjeti nekoliko predefiniranih grupa, ali i zadatke različitih korisnika koji su uneseni po defaultu u Camunda platformu, radi lakšeg učenja (John, Marry, Peter).

Odaberite **All Tasks** kako biste vidjeli sve zadatke.

Obzirom da ste prijavljeni kao **demo**, koji ima **administratorske ovlasti**, vidjet ćete sve zadatke koji su trenutno aktivni u Camunda platformi.

Dakle, možete se samostalno dodijeliti na zadatak **Potvrda narudžbe**; pritisnite na **claim** skroz desno.

The screenshot shows the Camunda Tasklist interface. On the left, a sidebar lists categories like 'My Tasks', 'My Group Tasks', 'Accounting', and 'John's Tasks'. Under 'My Tasks', several tasks are listed: 'Potvrda narudžbe', 'Assign Reviewer', 'Prepare Bank Transfer', 'Approve Invoice', and another 'Assign Reviewer'. The 'Potvrda narudžbe' task is selected and highlighted. On the right, a detailed view of this task is shown in a modal window. The title is 'Potvrda narudžbe' and it specifies 'webshop-order (v. v2.0)'. It contains fields for 'Invoice Amount' (10.99) and 'Invoice Number' (PSACE-5342). A question 'Želite li potvrditi narudžbu?' is displayed with a checkbox. At the bottom of the modal are 'Save' and 'Complete' buttons. The URL at the bottom of the page is localhost:8080/camunda/app/tasklist/default#/task=0613a25-b719-11ef-bc67-0242ac1f0002.

Pregled svih zadataka (All Tasks) u **Tasklist** aplikaciji

U sadržaju forme možete vidjeti polje **narudzba\_potvrdena** kojem ste dodijelili labelu **"želite li potvrditi narudžbu?"** i tip podatka **Boolean**.

Možete odabrati **checkbox** i pritisnuti na **complete** kako biste završili ovaj korisnički zadatak, ali ga i ne morate odabrati jer je polje **narudzba\_potvrdena** definirano kao **Boolean** tip podatka, što znači da je moguće unijeti i **true** i **false** vrijednosti.

Što će se dogoditi nakon što završite ovaj zadatak? 😊

Što ako ga završimo s **true** vrijednošću? A što ako ga završimo s **false** vrijednošću?

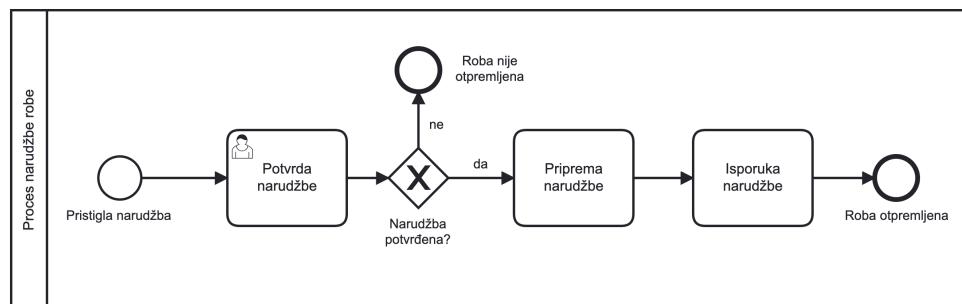
Instanca procesa završava u oba slučaja.

### 3.2.1 Procesne varijable i dodavanje **xor** skretnice

Vidjeli smo u modeliranju, da je uobičajeno da se nakon `User Taska` dodaje `xor` skretnica koja **određuje sljedeći korak** u procesu ovisno o rezultatu korisničkog zadatka, iako to ne mora uvijek biti slučaj.

Nadogradit ćemo proces dodavanjem `xor` skretnice koja će ovisno o rezultatu zadatka "Potvrda narudžbe" podijeliti proces na dva toka:

- ako je narudžba potvrđena, proces ide dalje na korak "Priprema narudžbe"
- ako narudžba nije potvrđena, proces završava



Dodavanje `xor` skretnice u definiciju procesa s alternativnim slijedom

Što je ovdje podatak/varijabla? 😊

U ovom slučaju, podatak je `narudzba_potvrdena` koji je definiran u formi korisničkog zadatka "Potvrda narudžbe". Ovaj podatak je **procesna varijabla** koja se pohranjuje u **procesnu instancu** i može se kasnije koristiti bilo gdje u procesu. Naziv procesne varijable smo definirali prilikom definiranja forme (`ID - Refers to the process variable name`).

Drugim riječima, procesna varijabla je **automatski pohranjena u procesnoj instanci** jednom kad se aktivnost "Potvrda narudžbe" završi.

Ono što moramo napraviti je definirati koristeći [Expression Language \(EL\)](#) izraze na izlaznim tokovima **XOR skretnice**.

Na izlaznim tokovima smo već napisali labele:

- da
- ne

**Međutim to nije dovoljno kod razvijamo procesne aplikacije!**

Osnovna sintaksa za **provjeru vrijednosti procesne varijable** je:

```
 ${varijabla == "vrijednost"}
```

U našem slučaju "vrijednost" je `true` ili `false`.

Dakle, na izlaznom toku koji ide prema "Priprema narudžbe" napišite izraz:

```
 ${narudzba_potvrdena == true}
```

Na izlaznom toku koji ide prema završnom događaju napišite izraz:

```
 ${narudzba_potvrdena == false}
```

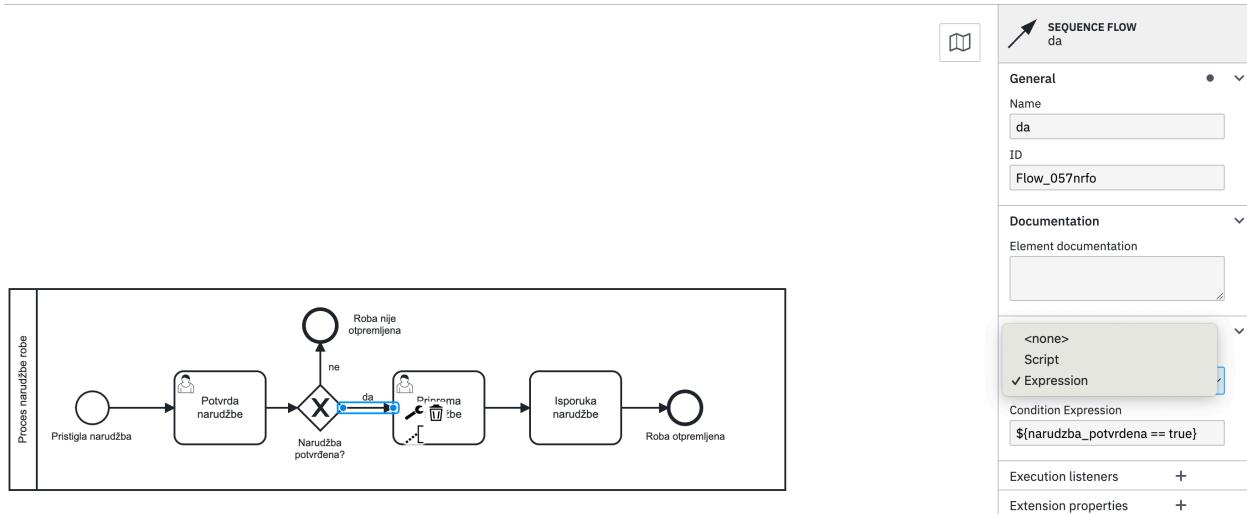
Alternativno, moguće je `Boolean` izraze napisati i skraćeno:

```
 ${narudzba_potvrdena}
```

odnosno ako nije potvrđena:

```
 ${!narudzba_potvrdena}
```

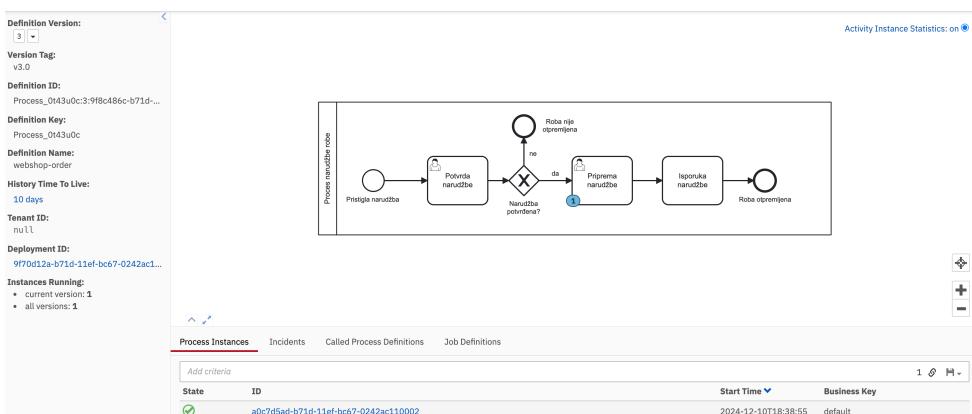
Odaberite strelice i definirajte `Condition Expression` za svaki izlazni tok (2):



Dodavanje izraza na izlazne tokove XOR skretnice (na izlazni tok "ne" dodan je izraz  
 `${!narudzba_potvrdena}` )

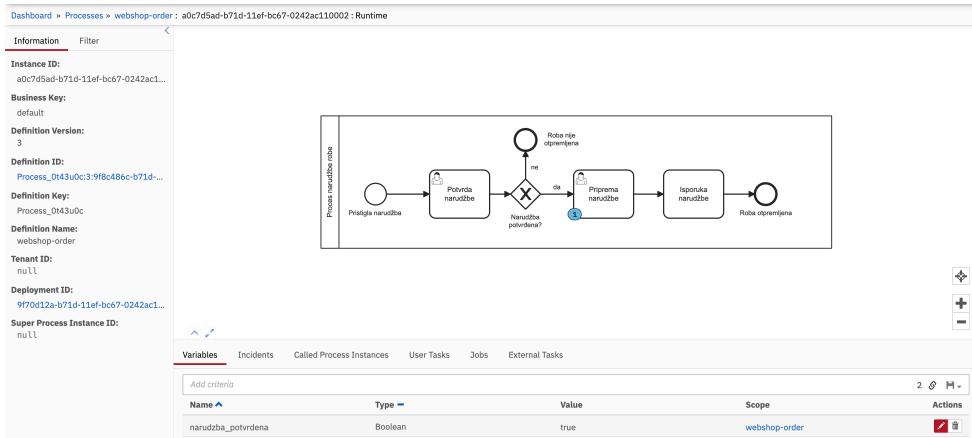
Ako sad ispunite formu i odaberete `true`, proces će ići prema aktivnosti `"Priprema narudžbe"`. Ako odaberete `false`, proces će završiti.

Provjerite rezultat u Camunda Cockpitu:



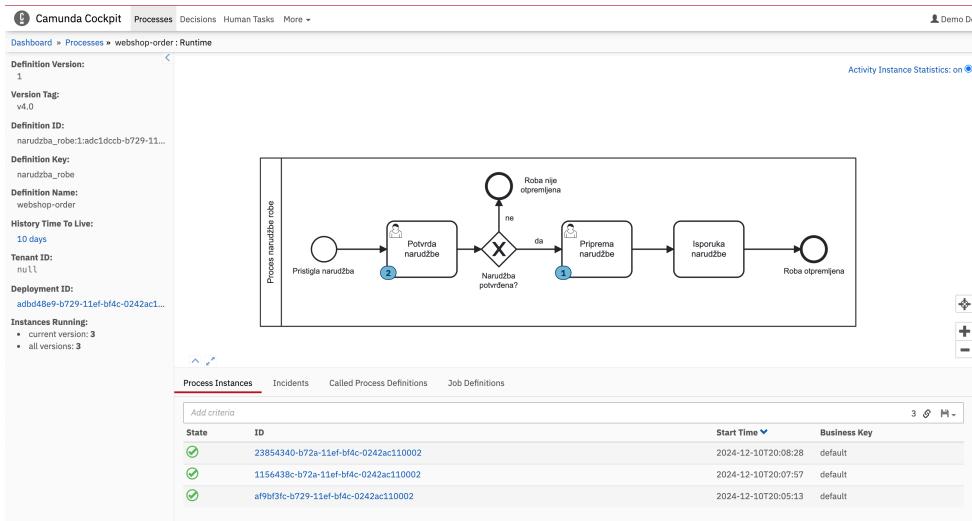
Token (1) se nalazi na aktivnosti `"Priprema narudžbe"`, što znači da **jedna aktivna instanca procesa čeka na ovom koraku**

Ako otvorite pregled procesne instance u Cockpitu, vidjet ćete da je procesna varijabla `narudzba_potvrdena` pohranjena u procesnu instancu i ima vrijednost `true`.



Pregled procesne instance s pohranjenom procesnom varijablom `narudzba_potvrdena`

Pokrenite još dvije instance ovog procesa kroz Modeler:



Prikaz 3 aktivne instance procesa, od kojih 2 čekaju na aktivnosti "Potvrda narudžbe", a jedna na "Priprema narudžbe"

### 3.2.2 Kako još možemo izraditi instance?

Osim ručnog pokretanja procesa preko Modelera, moguće je pokrenuti proces preko **REST API-ja**, ali i direktno iz **Camunda Tasklista**.

Do sad smo direktno izrađivali novu instancu procesa preko Modelera, iako je ovo praktično za testiranje, u stvarnom okruženju korisnici naravno neće imati pristup modeleru.

Problemi su sljedeći:

- Pristigla je narudžba - gdje su podaci o narudžbi? Kako ih unijeti?
- Kako krajnji korisnik može pokrenuti instancu procesa?

Procesne varijable možemo, osim kroz različite aktivnosti, **definirati i na početku**, kod započinjanja procesa. Proses koji modeliramo započinje primitkom narudžbe, logično je da onda i podaci o narudžbi budu procesne varijable.

Konkretno, podaci o narudžbi razlikovat će se u svakoj procesnoj instanci, samim tim je logično da ih ne definiramo unutar procesa (npr. u start eventu), već se unoše **prilikom pokretanja procesa**.

## 1. Način: Izrada procesne instance s varijablama preko Camunda Tasklista

- otvorite **Tasklist sučelje** i odaberite `start process` u gornjem desnom kutu

Camunda Tasklist      Keyboard Shortcuts Create task Start process Demo Demo Help

Odaberite `webshop-order` proces:

- unesite `Business Key` (proizvoljno): predstavlja jedinstveni ključ procesne instance (npr. u stvarnosti može biti ID narudžbe)
- dodajte varijable pritiskom na "Add a variable" i unesite neke podatke o narudžbi u obliku ključ:vrijednost parova

Start process: webshop-order

ⓘ You can set variables, using a generic form, by clicking the "Add a variable" link below.

Add a varia... +	Name	Type	Value
Remo... x	proizvod	String	Crna jakna
Remo... x	cijena	Double	150.25
Remo... x	kolicina	Integer	2

[Back](#) [Close](#) [Start](#)

### Pokretanje procesa s varijablama preko Tasklista

Dobit ćete poruku `"Process has been started."`

Provjerite procesne varijable pohranjene u procesnu instancu u **Cockpitu**.

## 2. Način: Izrada procesne instance s varijablama preko REST API-ja

Jednom kad se Camunda platforma pokrene, automatski se pokreće i REST API koji omogućuje komunikaciju s platformom preko HTTP protokola. REST API je dokumentiran i možete pronaći sve informacije na [sljedećoj poveznici](#).

Ako otvorite Modeler, vidjet ćete da je REST endpoint sljedeći:

`http://localhost:8080/engine-rest`

Otvorite **Postman** ili **Thunder Client**, možete poslati GET zahtjev na `http://localhost:8080/engine-rest/process-definition` kako biste dobili sve definicije procesa:

```

39     "name": "Invoice Receipt",
40     "version": 2,
41     "resource": "invoice.v2.bpmn",
42     "deploymentId": "9da07cc9-b728-11ef-bf4c-0242ac110002",
43     "diagram": null,
44     "suspended": false,
45     "tenantId": null,
46     "versionTag": "V2.0",
47     "historyTimeToLive": 45,
48     "startableInTasklist": true
49   },
50   {
51     "id": "narudzba_robe:1:adc1dccb-b729-11ef-bf4c-0242ac110002",
52     "key": "narudzba_robe",
53     "category": "http://bpmn.io/schema/bpmn",
54     "description": null,
55     "name": "webshop-order",
56     "version": 1,
57     "resource": "webshop-order.bpmn",
58     "deploymentId": "adbd48e9-b729-11ef-bf4c-0242ac110002",
59     "diagram": null,
60     "suspended": false,
61     "tenantId": null,
62     "versionTag": "v4.0",
63     "historyTimeToLive": 10,
64     "startableInTasklist": true
65   }
66 ]

```

Uočite proces `webshop-order` kao drugu vrijednost u JSON listi

Za pokretanje procesa, koristimo **POST metodu i endpoint** `http://localhost:8080/engine-rest/process-definition/key/<ProcessID>/start`, gdje je `<ProcessID>` ključ procesa, npr. `webshop-order` ili `narudzba_robe` - ovisno kako ste ga definirali u Modeleru (pogledati poglavlje 3.1.1).

<input type="checkbox"/> PARTICIPANT	Proces narudžbe robe
<b>General</b>	
Participant Name	Proces narudžbe robe
Participant ID	Participant_Omwnvv3
Process ID	narudzba_robe
Process name	webshop-order
Version tag	v4.0
<input checked="" type="checkbox"/> Executable	

Ključ procesa je `narudzba_robe`

**Endpoint za pokretanje procesa je:**

`http://localhost:8080/engine-rest/process-definition/key/narudzba_robe/start`

Međutim, kako bi pokrenuli proces s varijablama, moramo dodati i varijable u **tijelu HTTP zahtjeva**.

Varijable je potrebno omotati u JSON objekt, unutar ključa `variables`.

Npr. sljedeći JSON objekt započinje instancu procesa `narudzba_robe` s varijablama `proizvod`, `cijena` i `kolicina`:

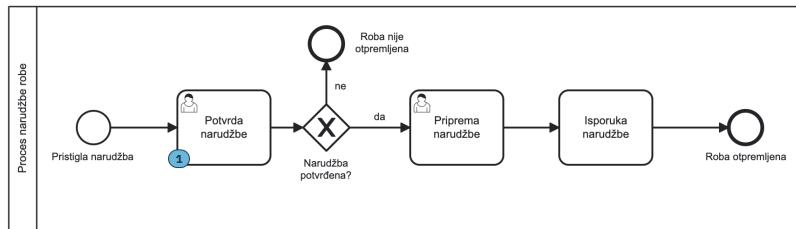
```
{  
    "variables": {  
        "proizvod": {  
            // ključ varijable  
            "value": "Majica", // vrijednost varijable  
            "type": "String" // tip varijable  
        },  
        "cijena": {  
            "value": 70,  
            "type": "Double"  
        },  
        "kolicina": {  
            "value": 2,  
            "type": "Integer"  
        }  
    }  
}
```

The screenshot shows a REST client interface with the following details:

- URL:** http://localhost:8080/engine-rest/process-definition/key/narudzba\_robe/start
- Method:** POST
- Body:** JSON payload containing the variables object from the previous code snippet.
- Response:** 200 OK, 382 ms, 505 B. The response body is a JSON object representing the created process instance, including its ID, definition ID, and links.

Izrada procesne instance s varijablama preko REST API-ja

Provjerite procesnu instancu i pohranjene varijable u **Cockpitu**.



Variables	Incidents	Called Process Instances	User Tasks	Jobs	External Tasks
Add criteria					
Name	Type	Value	Scope	Actions	
cijena	Double	70	webshop-order		
kolicina	Integer	2	webshop-order		
proizvod	String	Majica	webshop-order		

Pregled procesne instance definirane preko REST API-ja i njezinih pohranjenih varijabli

# 4. Obrada vrijednosti procesnih varijabli u procesu

Sad kad smo dodali mogućnost unosa podataka u proces preko **Tasklista i REST API-ja**, možemo **iskoristiti te podatke u procesu**.

Procesne varijable možemo dohvaćati na jednak način, bez obzira na način unosa, a to je sintaksom:

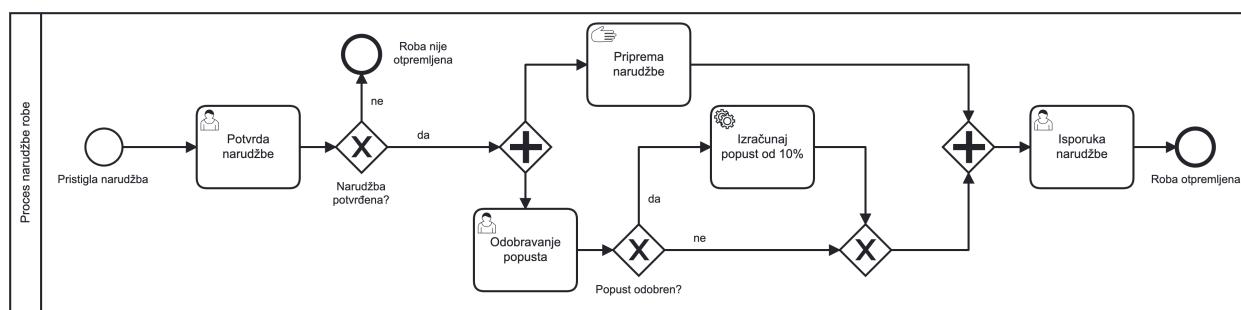
```
 ${naziv_procesne_varijable} .
```

Nadogradit ćemo proces `narudzba_robe` tako da se na temelju unesenih podataka o narudžbi, izračuna ukupna cijena narudžbe. Slijed procesa se sad paralelno dijeli na `"Priprema narudžbe"` i novi User Task - `"Odobravanje popusta od 10%"`. Korisnik može odobriti ili odbiti popust na narudžbu, a prilikom odobravanja/odbijanja mora unijeti i svoje ime i prezime. Ako korisnik odobri popust, isti se mora izračunati i pohraniti u procesnu varijablu. Ako korisnik odbije popust, proces nastavlja na jednak način ali bez izračuna popusta.

Koje atribute ćemo koristiti za ovaj nadograđeni proces?

- `"Potvrda narudžbe"` - **User Task**
- `"Odobravanje popusta"` - **User Task**
- `"Izračunaj popust od 10%"` - **Service Task**
- `"Priprema narudžbe"` - **Manual Task**
- `"Isporuka narudžbe"` - **User Task** - čisto da nam instance ne završi odmah, inače bi bio *manual task* ili *potproces*

Nakon `"Odobravanje popusta"` želimo izračunati ukupnu cijenu narudžbe i pohraniti ju u novu procesnu varijablu `ukupna_cijena`. Ova aktivnost ide u `XOR split` skretnicu `"Popust odobren?"` koja ovisno o rezultatu ide na `"Izračunaj popust od 10%"` ili direktno u `AND merge` skretnicu.



Prvo ćemo definirati formu za `Odobravanje popusta`. Odaberite `Generated Task Forms` i dodajte polja:

- `popust_odobren` - tip podatka `Boolean`, labela: `"Želite li odobriti popust od 10%?"`
- `djelatnik_ime` - tip podatka `String`, labela: `"Molimo unesite vaše ime"`
- `djelatnik_prezime` - tip podatka `String`, labela: `"Molimo unesite vaše prezime"`

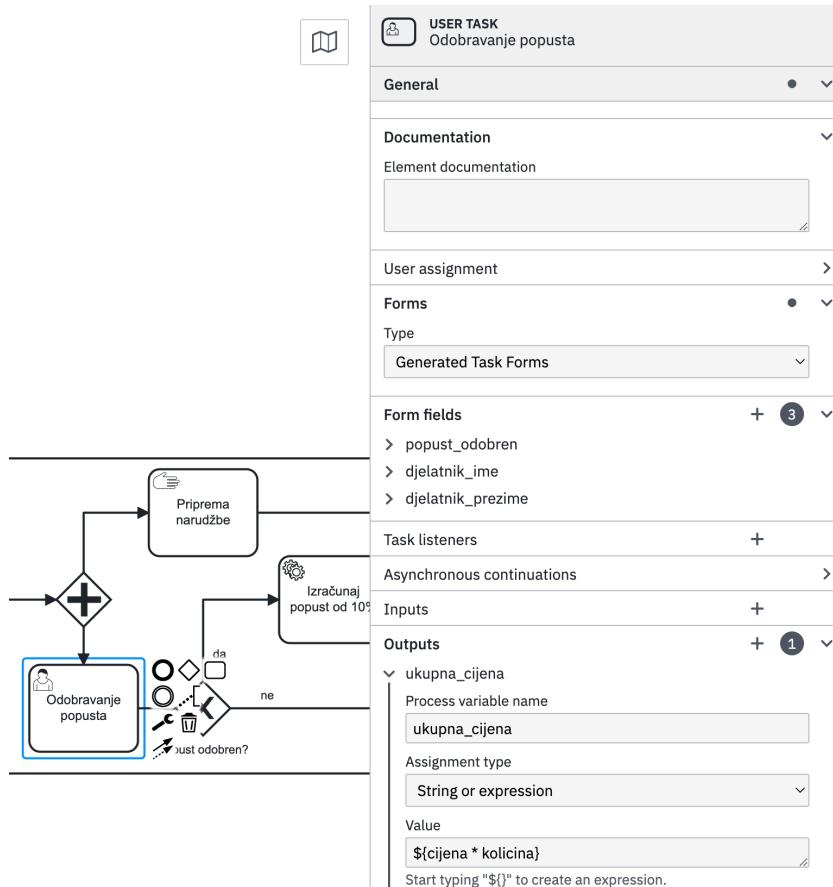
**Gotovo svaka aktivnost može kao rezultat svog izvršavanja pohraniti neku procesnu varijablu.**

Moguće je iskoristiti vrijednosti postojećih varijabli te unutar `Expression` izraza izračunati nove vrijednosti.

**Mi želimo izračunati ukupnu cijenu narudžbe** koristeći procesne varijable `cijena` i `kolicina` te pohraniti rezultat u novu varijablu `ukupna_cijena`.

Odaberite `Outputs` i dodajte novu varijablu `ukupna_cijena`. Odaberite `String or expression` te kao vrijednost pohranite:

```
 ${cijena * kolicina}
```



Dodavanje 3 polja u Form fields korisničkog zadatka "Odobravanje popusta" te dodavanje izračunate procesne varijable `ukupna_cijena` kao izlaznu vrijednost (**Outputs**) ovog zadatka

Na izlaznom toku `Da` dodajte izraz:

```
 ${popust_odobren}
```

Na izlaznom toku `Ne` dodajte izraz:

```
 ${!popust_odobren}
```

Izlazni tok `Da` ide na aktivnost "Izračunaj popust od 10%", a izlazni tok `Ne` ide direktno u AND merge skretnicu.

Na "Izračunaj popust od 10%" aktivnosti ćemo odabrati **servisni zadatak**. Servisni zadatak je aktivnost koja se izvršava automatski, bez korisničke interakcije, i može izvršiti neki posao, npr. izračunati popust. Odaberite jednostavni izraz:

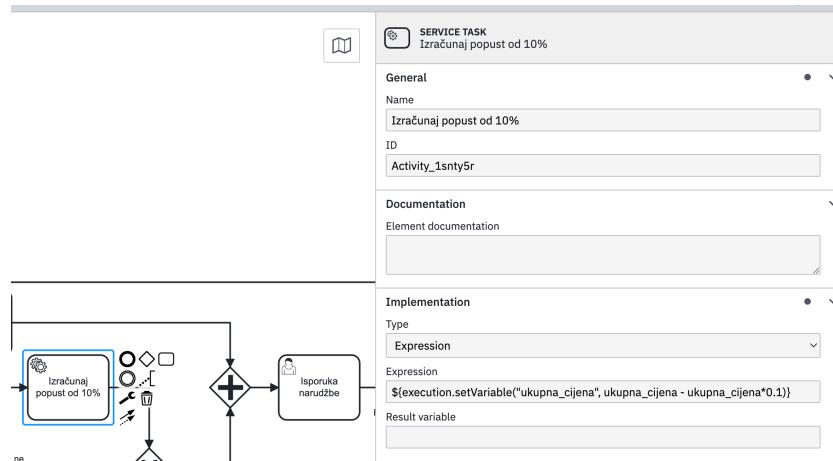
Implementation` -> `Type` -> `Expression

Sad možemo definirati izraz koji će promijeniti vrijednost procesne varijable ukupna\_cijena:

```
`${execution.setVariable("varijabla", vrijednost)}
```

U našem slučaju:

```
`${execution.setVariable("ukupna_cijena", ukupna_cijena - ukupna_cijena*0.1)}
```



Definiranje izraza za izračun popusta od 10% na servisnom zadatku "Izračunaj popust od 10%"

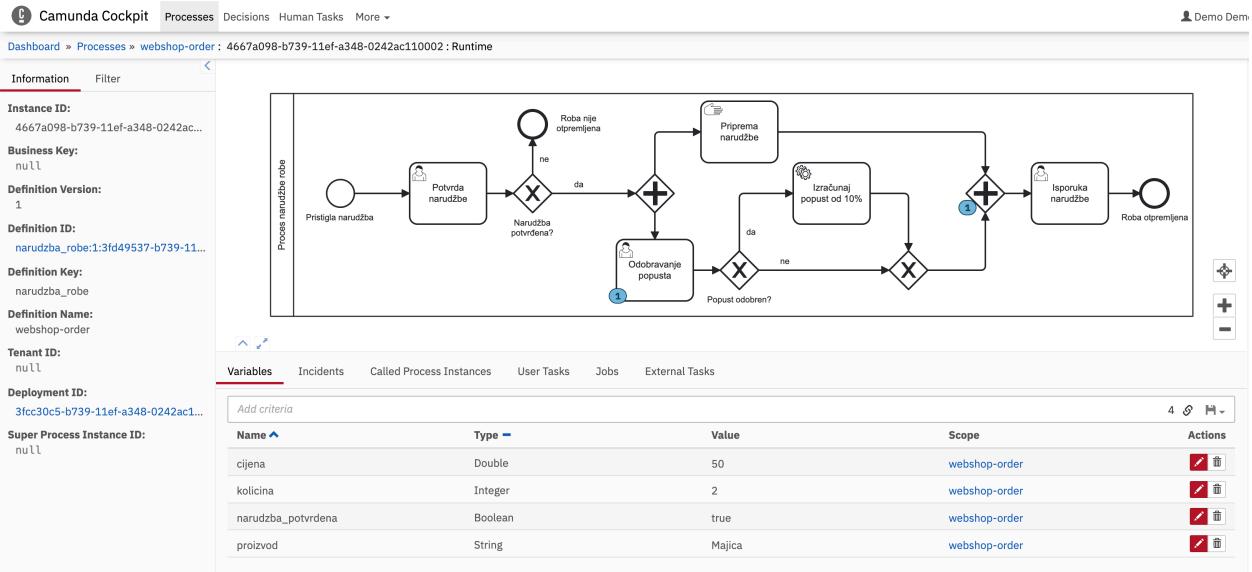
To je to! **Redployajte novu verziju procesa i pokrenite novu instancu procesa kroz REST API ili Tasklist.** Dodajte početne procesne varijable: `proizvod`, `cijena`, `kolicina` i pratite tijek procesa kroz **Cockpit**.

The screenshot shows the Camunda Cockpit interface. On the left, there's a sidebar with 'Information' selected, displaying process details like Instance ID, Business Key, Definition Version, and so on. The main area shows the process definition with its states and transitions. At the bottom, a 'Variables' table lists initial values for 'cijena', 'kolicina', and 'proizvod'.

Name	Type	Value	Scope	Actions
cijena	Double	50	webshop-order	
kolicina	Integer	2	webshop-order	
proizvod	String	Majica	webshop-order	

Početno stanje instance procesa s varijablama `proizvod`, `cijena` i `kolicina`, čekanje na aktivnost "Potvrda narudžbe"

Nakon potvrde narudžbe, paralelno se izvršavaju aktivnosti "Priprema narudžbe" i "Odobravanje popusta". Međutim, proces čeka na egzekuciju "Odobravanje popusta" budući da je to User Task.



Prikaz aktivne instance procesa s tokenom na aktivnosti "Odobravanje popusta" i AND merge skretnici budući da se manualni taskovi preskaču

Otvaramo **Tasklist** i odabiremo zadatku "odobravanje popusta". Unosimo podatke i odobravamo popust.

Heads-up – it is time to try out Camunda 8. The last release of Camunda 7 CE will be in October 2025. If you have any more questions, please get in touch with us in the forum.

Keyboard Shortcuts Create task Start process Demo Demo

My Tasks (3) Filter Tasks 3 ↻ ↻ Add Comment +

**Odobravanje popusta**

Created 3 minutes ago Demo Demo 50

Set follow-up date Set due date Add groups Demo Demo

Form History Diagram Description

**Odobravate li popust od 10%?**

**Molimo unesite vaše ime**  
Marko

**Molimo unesite vaše prezime**  
Marić

Save Complete

Generirana forma za "Odobravanje popusta" s unesenim podacima za procesne varijable:

popust\_odobren, djetatnik\_ime i djetatnik\_prezime

Otvorite **Cockpit** i pogledajte stanje procesne instance i unesenih varijabli. Vidjet ćete da se izračunao popust od 10% i pregazio vrijednost procesne varijable ukupna\_cijena, koja je bila 100. Token se sada nalazi na aktivnosti "Isporuka narudžbe", kako se instanca ne bi završila odmah (premda nismo definirali kako dalje).

Dashboard > Processes > webshop-order : 4667a098-b739-11ef-a348-0242ac110002 : Runtime

**Information** Filter

**Instance ID:** 4667a098-b739-11ef-a348-0242ac...

**Business Key:** null

**Definition Version:** 1

**Definition ID:** narudzba\_robe:1:3fd49537-b739-11...

**Definition Key:** narudzba\_robe

**Definition Name:** webshop-order

**Tenant ID:** null

**Deployment ID:** 3fc30c5-b739-11ef-a348-0242ac1...

**Super Process Instance ID:** null

```

graph LR
    Start(( )) --> Potvrda[Potvrda narudžbe]
    Potvrda --> Decision{Narudžba potvrđena?}
    Decision -- ne --> Roba[Roba nije otpremljena]
    Decision -- da --> Prikrepa[Priprema narudžbe]
    Prikrepa --> Izracunaj[Izračunaj popust od 10%]
    Izracunaj --> Odobravanje{Odobravanje popusta}
    Odobravanje -- da --> Popust{Popust odobren?}
    Popust -- da --> Isporuka[Isporuka narudžbe]
    Popust -- ne --> Roba
    Isporuka --> RobaOtpremljena((Roba otpremljena))
    
```

**Variables**

Name	Type	Value	Scope	Actions
cijena	Double	50	webshop-order	[edit] [history]
djelatnik_ime	String	Marko	webshop-order	[edit] [history]
djelatnik_prezime	String	Marić	webshop-order	[edit] [history]
kolicina	Integer	2	webshop-order	[edit] [history]
narudzba_potvrdena	Boolean	true	webshop-order	[edit] [history]
popust_odobren	Boolean	true	webshop-order	[edit] [history]
proizvod	String	Majica	webshop-order	[edit] [history]
ukupna_cijena	Double	90	webshop-order	[edit] [history]

Prikaz aktivne instance procesa s tokenom na aktivnosti "Isporuka narudžbe" nakon izračuna popusta od 10% i pohranjenih varijabli.

# Samostalni zadatak za Vježbu 5

---

Modelirajte jednostavni proces prijave studentske prakse na Fakultetu informatike u Puli. Postoje 3 sudionika u procesu prakse:

1. **Student**
2. **Poslodavac**
3. **Profesor**

Proces započinje kod studenta odabirom zadatka za praksu. Student ispunjava web formu gdje unosi svoje ime, prezime, JMBAG i šifru zadatka (izmislite podatke).

Sljedeći korak je odobravanje prakse od strane profesora. Profesor pregledava podatke studenta i šifru zadatka u web sučelju, a nakon toga odobrava ili odbija prijavu. Ako prijava nije prihvaćena, proces se vraća na studenta i njegovu aktivnost ispunjavanja web forme. Ako profesor prihvati prijavu, proces se nastavlja kod poslodavca. Poslodavac provodi intervju sa studentom, a nakon toga odlučuje hoće li ga prihvati ili odbiti. Ako ga odbije, proces se ponovno vraća na studenta i njegov unos podataka. Ako ga prihvati, proces ide prema studentu koji sad mora unijeti kratak opis zadatka, datum izvođenja prakse te ime i prezime mentora koji mu je dodijeljen te istovremeno prema profesoru kojeg se samo obavještava. Nakon izvršavanja tih paralelnih aktivnosti, proces se završava.

Nakon što ste modelirali proces, implementirajte procesnu aplikaciju u **Camundi 7**:

- Dodajte definirane korisničke aktivnosti i korespondirajuće forme
- Definirajte procesne varijable i njihove vrijednosti
- Definirajte skretnice i uvjete na izlaznim tokovima
- Obavještavanje sudionika procesa ne implementirate

Predajete isključivo `.bpmn` datoteku procesa i aplikacije definirane u Camunda Modeleru.