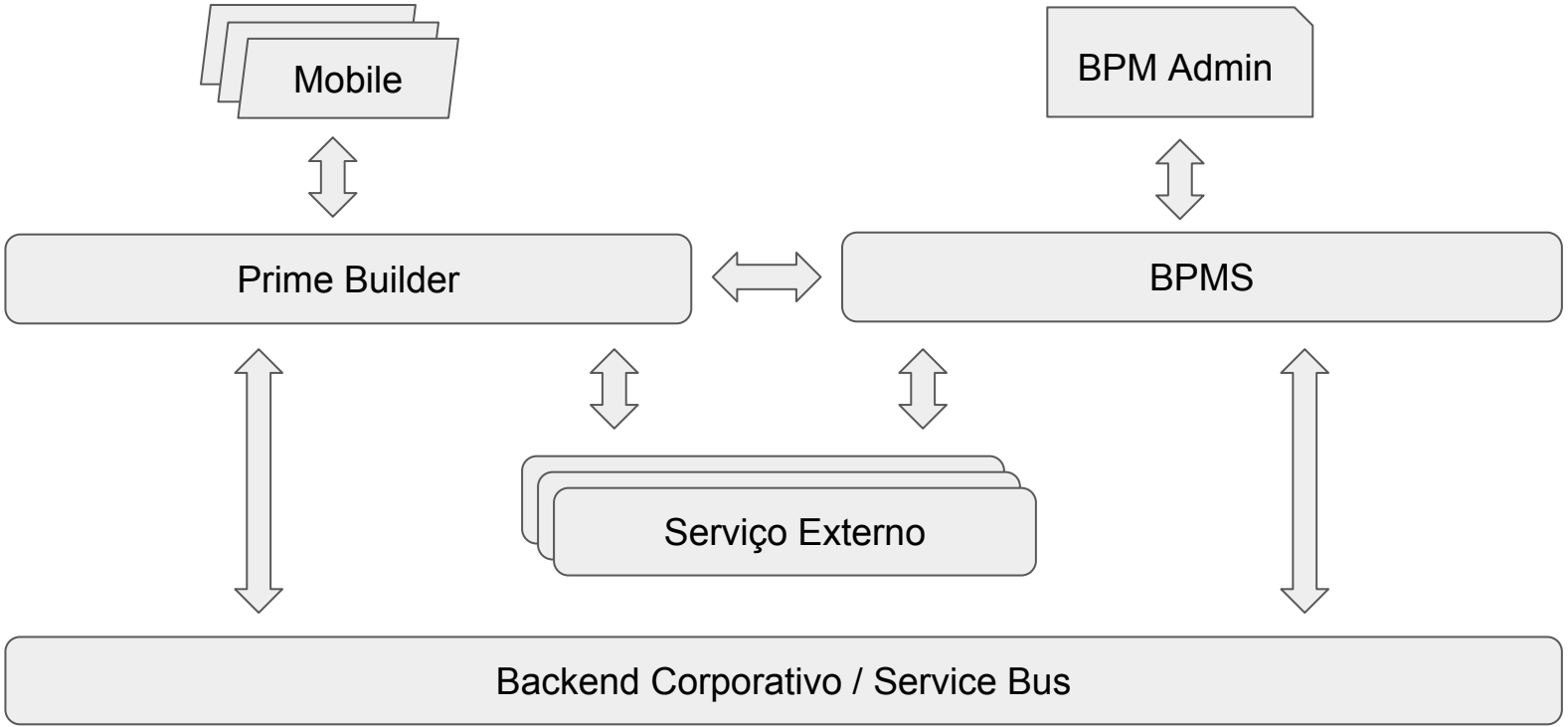
Proposta: PrimeBuilder + BPM (PB++)

Premissas

1. PrimeBuilder *as is*. Nenhuma alteração na versão atual.
2. Acoplamento fraco: as interações PB-BPMS e BPMS-PB devem ser *stateless* e feitas sem o compartilhamento de nenhum recurso.
3. Desenho dos processos no padrão BPMN
4. Uso de um editor gráfico de mercado que exporte os modelos num padrão aberto (XPDL, BPMN2 ou outro dialeto xml) que possa ser convertido para um formato contendo as informações necessárias à interpretação dos processos.

Arquitetura: visão geral



PrimeBuilder + BPMS

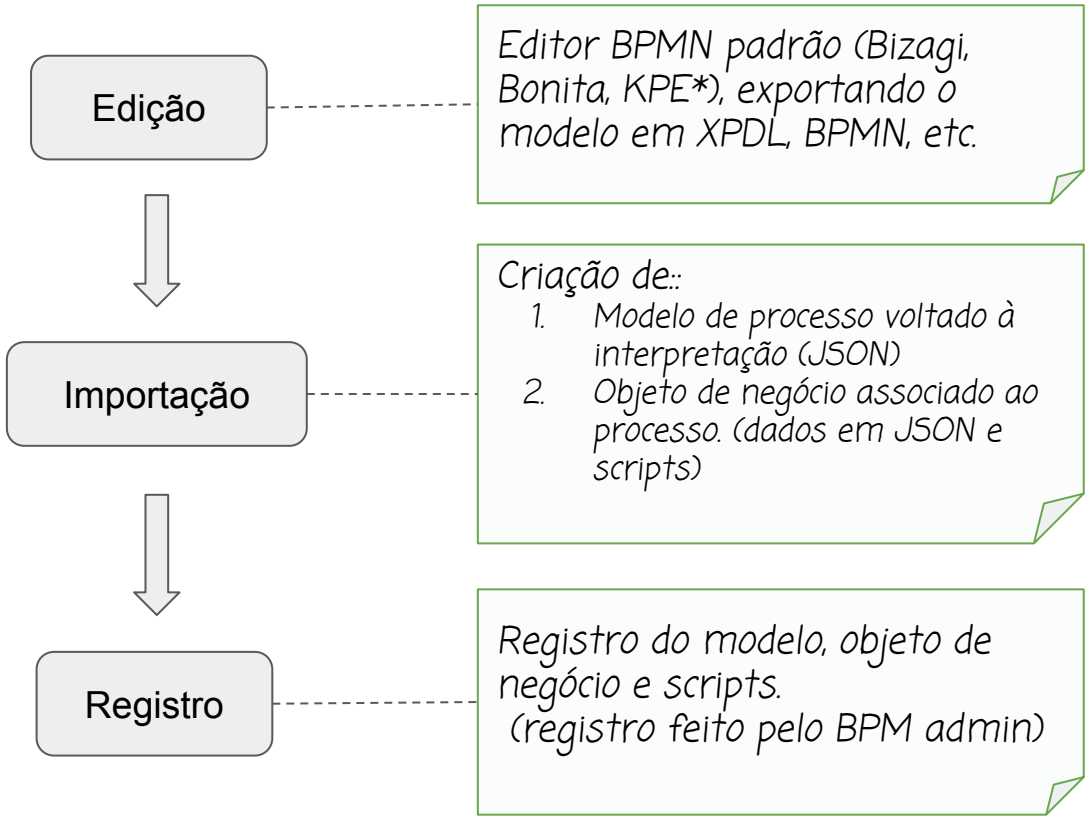
- As tarefas de usuário do BPMS são executadas através do PrimeBuilder
  - Quando uma tarefa é habilitada no BPMS, é disparado um serviço que cria uma OS no PrimeBuilder, para o usuário associado ao papel definido no processo.
  - Quando a OS é concluída no PrimeBuilder este dispara um serviço que sinaliza a conclusão.
  - A conclusão da OS é tratada pelo BPMS como a execução de uma tarefa, que por sua vez habilita outras tarefas ou finaliza o processo.
- PrimeBuilder e BPMS podem acessar serviços externos e o backend corporativo através de web services.
- A única interface de usuário ‘direta’ com o BPMS é a do administrador do BPMS:
  - Atualização de modelos
  - Consultas e relatórios

BPMS

Proposta

- BPMS dedicado, a ser implementado como uma extensão natural do PrimeBuilder.
- Interação com o PrimeBuilder via web services, como cliente e também como servidor.
- Implementação baseada em micro serviços.
- Voltado unicamente à interpretação (ou orquestração) dos processos:
    - Toda parte de UI relativa aos processos é feita pelo PrimeBuilder (mobile ou html5)
    - O gerenciamento das listas de tarefas e controle das mesmas é feita pelo PrimeBuilder.
    - A persistência feita pelo BPMS se restringe aos dados necessários a sua própria operação.

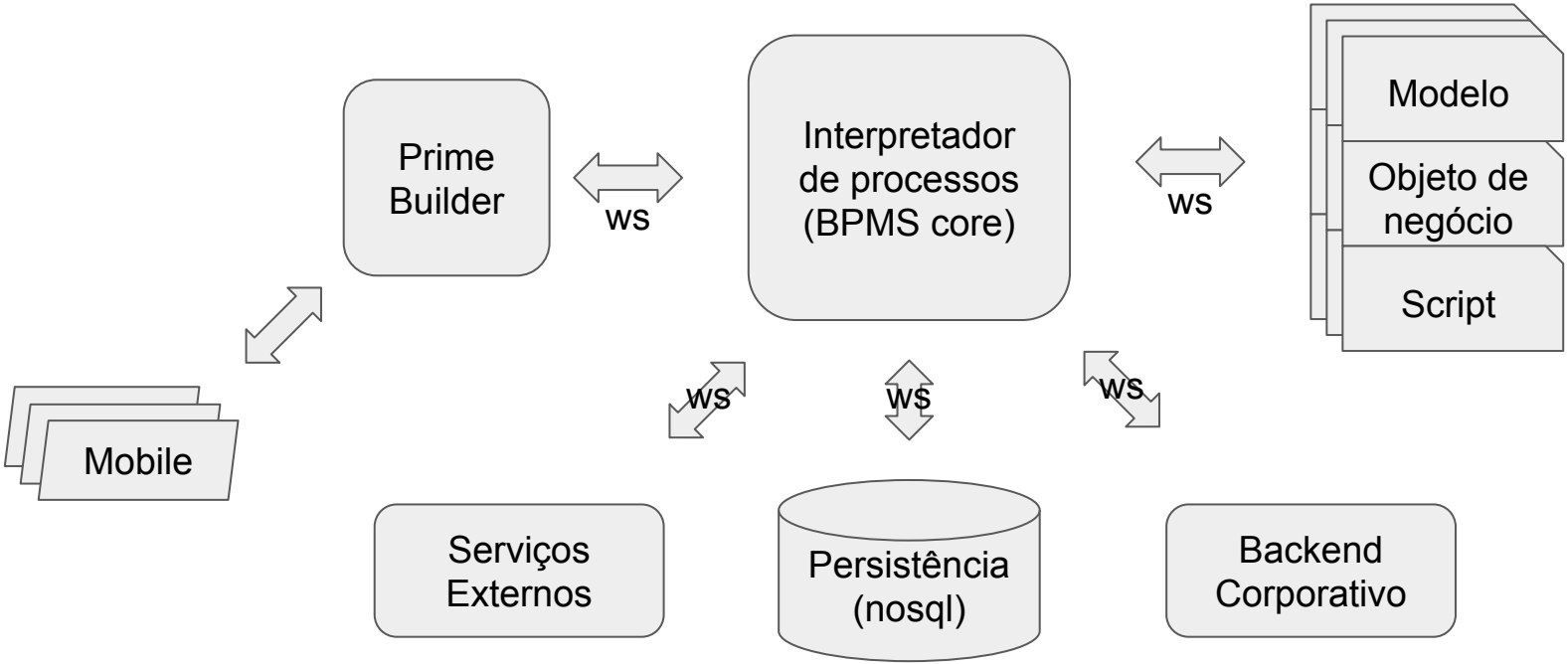
Criação do modelo de processo BPM



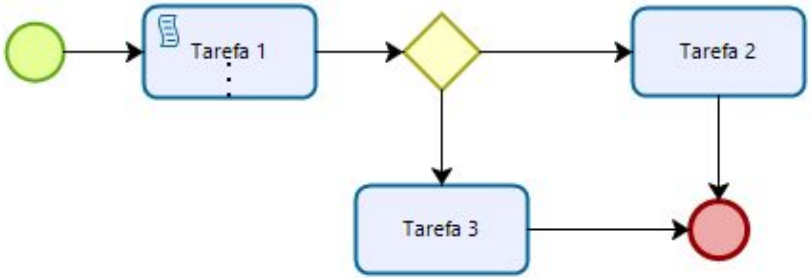
Importação

- Feita por uma ferramenta dedicada (acoplada ou não ao BPMS)
- Em dois passos:
  a. Conversão de XPDL ou BPMN para uma representação JSON voltada à interpretação do processo.
  b. Criação do ‘objeto de negócio’ associado ao processo
     - Definição dos dados (variáveis do processo)
     - Definição dos scripts associados às atividades do processo
     - Dados e scripts são
       - independentes daqueles definidos no mobile
       - Compartilháveis entre as etapas

Interpretação dos processos



Modelo de Processo



- Cada etapa do processo é representada por um objeto que a descreve.

Ex:

```
{    "id": "id-1",
     "name": "Tarefa 1",
     "type": "task",
     "role": "supervisor",
     "succ": "id-2", "pred": "id-0"
}
```

Objeto de negócio

- Define os dados associados ao processo (um objeto para cada modelo de processo).
- Os dados podem ser manipulados pelos scripts associados às atividades.
- Cada instância de processo é associada a uma instância do objeto de negócio.

Exemplo:

```
{    “instanceId”: “ab123”,

     “nome”: String,

     “endereco”: String,

     “valor”: Float,

     “status”: String,

}
```

Scripts

- Define as funções de tratamento dos eventos associados às atividades do processo.
- Eventos:
  - onEnable
  - onCommit
  - onUndo
- Usa os dados definidos no objeto de negócio como variáveis do processo.
- Linguagem: Javascript é a candidata natural.
- A ferramenta de importação pode gerar a primeira versão dos scripts contendo apenas funções vazias.
- Podem fazer uso de funções pré-definidas específicas

Processos e OS’s

- Um processo BPM é formado por várias atividades e gateways
- Uma atividade BPM pode ser
  - Executada por um usuário
  - Associada à execução automática de um script
  - Associada a um evento
  - Associada à recepção de uma mensagem
- Cada atividade executada por um usuário vai corresponder a um fluxo no PB
  - Um processo BPM pode ter associados vários fluxos PB
  - Um fluxo PB pode ser compartilhado entre os processos BPM

Criação de uma instância de processo BPM

- Automática: uma aplicação da organização cliente aciona o serviço do BPMS para a criação da instância, passando o objeto de negócio a ser associado à mesma.
- Pelo usuário: o usuário executa a OS associada à tarefa inicial do processo BPM (a OS deve ser previamente enviada ao seu dispositivo).

POC

1. Primeira Etapa
    a. Importação e interpretação de um subset(*) do BPMN a partir do XPDL gerado pelo Bizagi (interpretador como microservice)
    b. Stub para persistência dos modelos e instâncias (como microservice)
    c. Ações de usuário simuladas em batch via linha de comando
    d. Resultados na console
2. Segunda Etapa
    a. Fluxo BPM exemplo
    b. Interfaces/integração PB -> BPMS e BPMS -> PB
    c. Fluxos PB para as atividades previstas no(s) processo(s) de teste

Atividades primeira etapa

1. Importação 24h (Bizagi, XPDL)
2. Verificação de consistência do processo 8h
3. Interpretador 24h
4. Mockup objeto de negócio + scripts 4h
5. Stub da persistência 4h
6. Código de testes 4h

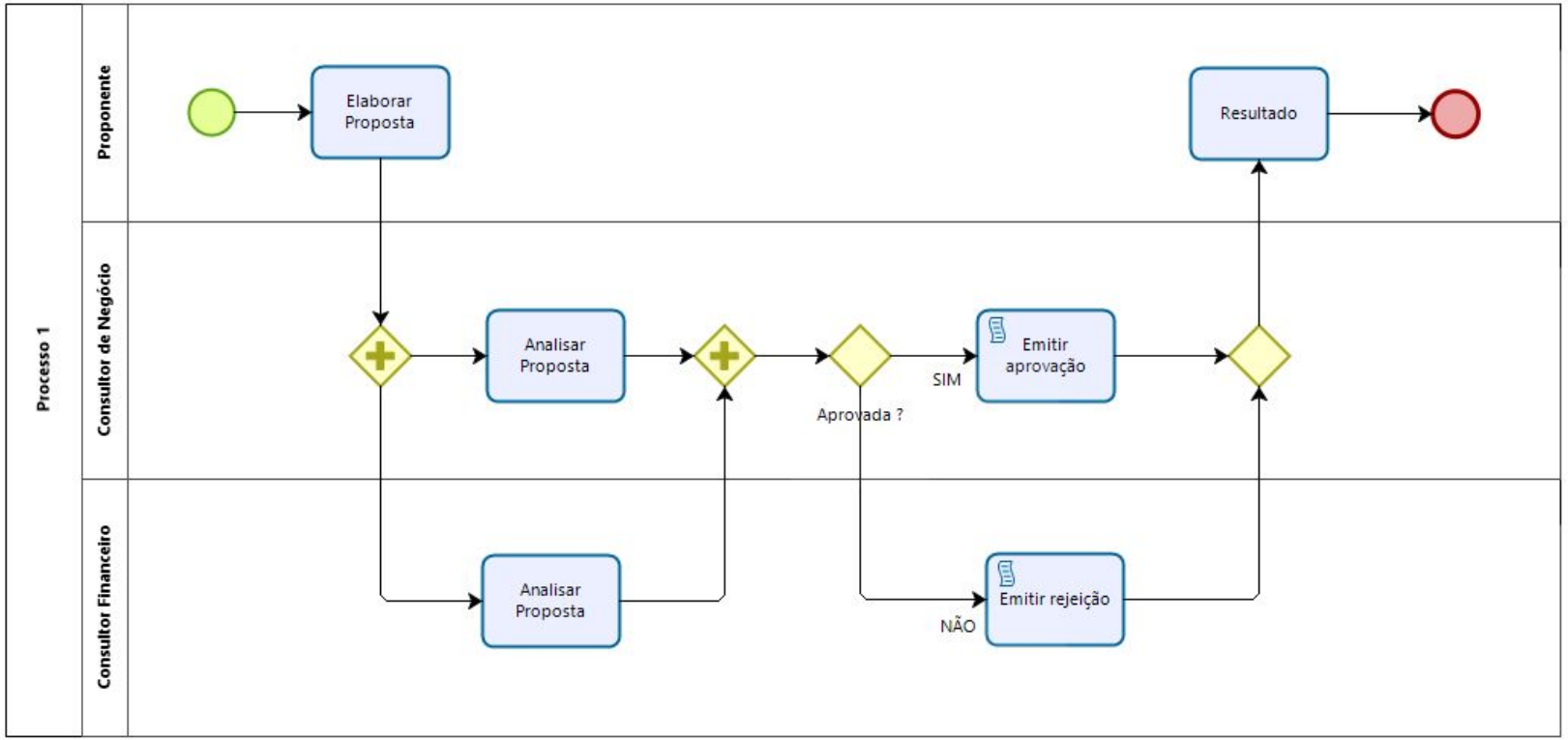
POC - Parte 1

- Grupos de funcionalidades implementadas
  - Importação do xpdl
    - xpdlImporter.js
  - Geração do modelo a ser interpretado e stub dos scripts em arquivos separados
    - converter.js e translator.js
  - Registro do modelo + script
    - modelManager.js
  - Interpretação
    - instanceManager.js e interpreter.js

Tasks, gateways e eventos tratados

- Tasks: UserTask, ScriptTask
- Gateways: Exclusivo e Inclusivo (internamente como pares Fork-Join)
- Eventos: Start e End
- Associação de scripts:
  - Apenas tasks, gateways e eventos que tenham um nome podem ter scripts associados.

Exemplo de modelo de processo usado nos testes



Scripts

1. A importação do modelo xpdl gera um arquivo javascript contendo o boneco da função de script para cada task, gateway ou evento do modelo original que tenha um nome.
2. O usuário poderá editar esse arquivo colocando as funcionalidades desejadas, sem alterar a estrutura geral do arquivo
    a. Os nomes e parâmetros originais das funções geradas devem ser mantidos.
    b. O corpo das funções poderá ser alterado.
    c. Funções auxiliares, se necessário podem ser inseridas.
    d. Variáveis globais podem ser criadas mas não são persistidas.
3. Além das funções associadas a tasks, gateways e eventos, são geradas:
    a. Função onStart() a ser executada a cada criação de instância de processo
    b. Função onFinish() a ser executada ao encerramento da instância.

Scripts

- Parâmetros
  - Todas as funções do script geradas pela ferramenta têm um único parâmetro:
    - instanceContext: objeto que contém
      - Os métodos getPath() e setPath() a ser usados no tratamento do caminho a seguir no caso do gateway Exclusive Fork.
      - Atributo instanceData, contendo os dados da instância tratados durante a execução das atividades relacionadas à mesma.
- Dados da Instância
  - Passados como parâmetro à criação da instância
  - Atualizados durante a execução das tarefas e dos scripts associados
- Bibliotecas auxiliares
  - Além dos métodos básicos oferecidos pelo instanceContext, será possível usar outras bibliotecas que forneçam as funções necessárias ao uso prático do BPM.

Funcionalidades ‘exportadas’

- Administrativas
  - Registro, ativação, desativação de modelos de processos
  - Consultas aos modelos de processo e instâncias
- Operacionais
  - Criação de uma instância: disparada por um ‘evento externo’ (tipicamente gerado pelo PB)
  - Consulta a lista de tarefas associada a um modelo de processo
  - Execução de uma tarefa (PB)