DISTRIBUITED DATA

(DDP)

Luiz Quelves
Ronald Pérez

Contenido

- Definición y funcionalidades
- Arquitectura
- Publicaciones
- Subscripciones
- Conclusiones

DDP | Definición

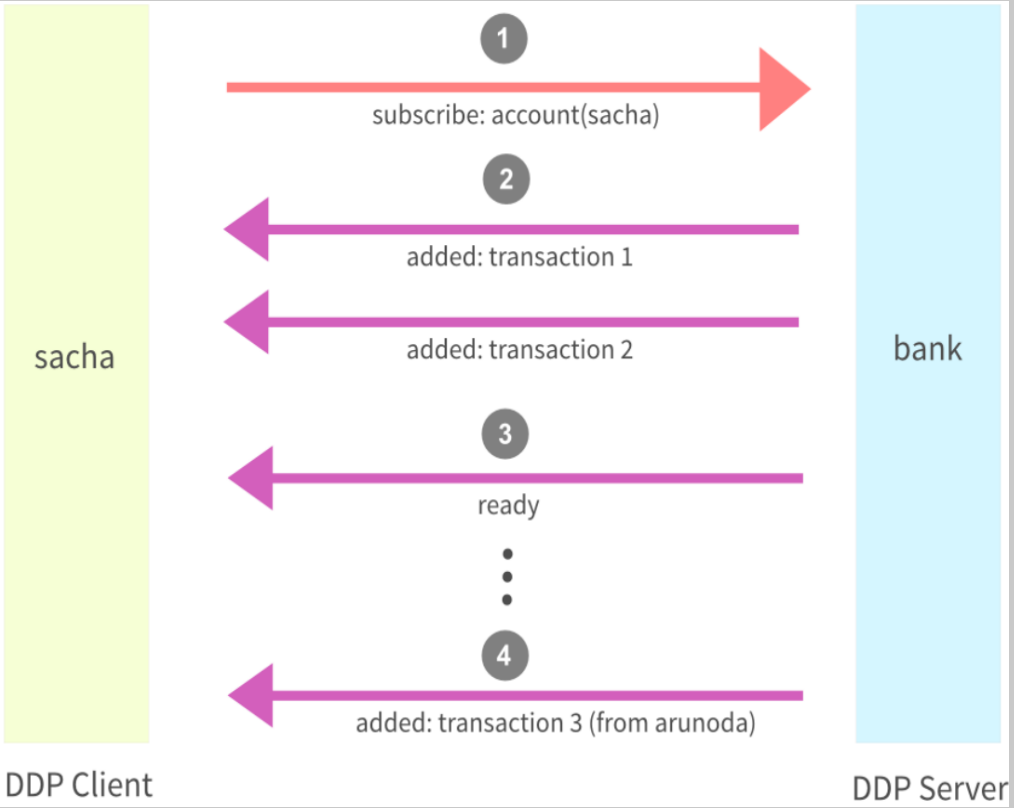
Es un protocolo que permite consultar la bases de datos del lado del servidor, enviando resultados al cliente y publicando los cambios en la base de datos

DDP | Funcionalidades

- Maneja llamados a procedimientos remotos
  - Se pueden invocar métodos de el servidor y obtener datos de regreso.

- Gestiona datos.
  - Se puede suscribir en tiempo real a datos y notificaciones
    - Tipos de notificaciones
      - Added
      - Changed
      - Removed

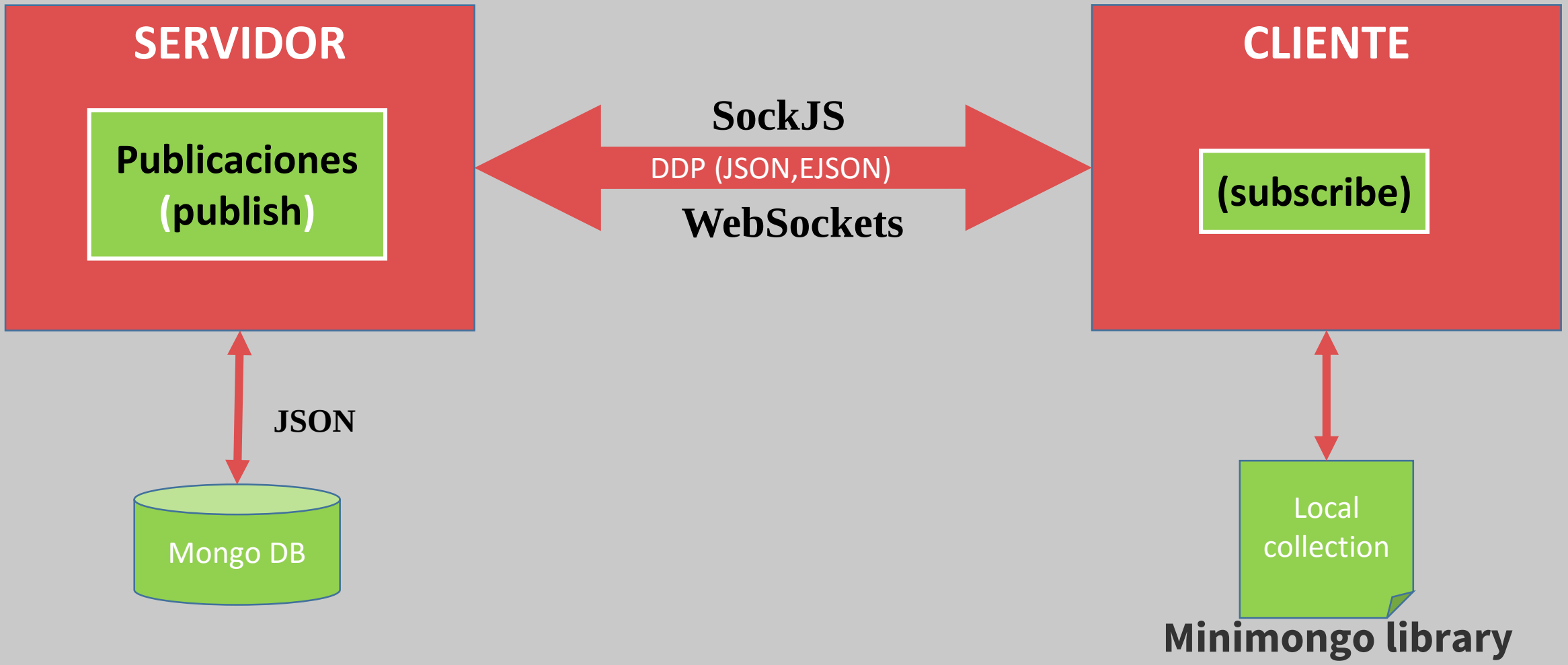
Ejemplo de comunicación




```
1.{"msg": "sub", id: "random-id-2", "name": "account", "params": ["sacha"]}
2.{"msg": "added", "collection": "transactions", "id": "record-1", "fields": {"amount": "50
USD", "from": "tom"}}
  {"msg": "added", "collection": "transactions", "id": "record-2", "fields": {"amount": "15
0USD", "from": "chris"}}
3.{"msg": "ready": "subs": ["random-id-2"]}
4.{"msg": "added", "collection": "transactions", "id": "record-3", "fields": {"amount": "10
00USD", "from": "arunoda"}}
```

Archivo JSON

DDP | Arquitectura




```
Todos = new Mongo.Collection('Todos');
```

```
Todos = new Mongo.Collection('Todos');
```

DDP | cliente - servidor

Server

Meteor.publish

- Pasa una función para publicar datos
- Database, IMAP, forklifts warehouse

Client

Meteor.subscribe

- Conectarse a una publicación
- Recibe datos y escucha actualizaciones

Publicaciones

- Es una API en el servidor que construye un grupo de datos para enviar al cliente.
- Utilizan **Meteor.publish()**
- Las publicaciones toman dos parámetros:
  - Contexto **this**
  - Argumentos para la publicación (Meteor.subscribe)
- No es compatible con funciones Arrow

Publicaciones | estructura

Cursor mongo

Nombre de la publicación

```
Meteor.publish('lists.public', function() {
  return Lists.find({
    userId: {$exists: false}
  }, {
    fields: Lists.publicFields
  });
});
```

Filtro de campos

```
// In the file where Lists is defined
Lists.publicFields = {
  name: 1,
  incompleteCount: 1,
  userId: 1
};
```

Publicaciones | userId

Argumento

Bad

```
Meteor.publish('list', function (listId) {
  check(listId, String);

  const list = Lists.findOne(listId);

  if (list.userId !== this.userId) {
    throw new Meteor.Error('list.unauthorized',
      'This list doesn\'t belong to you.');
  }

  return Lists.find(listId, {
    fields: {
      name: 1,
      incompleteCount: 1,
      userId: 1
    }
  });
});
```

Good

```
Meteor.publish('list', function (listId) {
  check(listId, String);

  return Lists.find({
    _id: listId,
    userId: this.userId
  }, {
    fields: {
      name: 1,
      incompleteCount: 1,
      userId: 1
    }
  });
});
```

```
Meteor.subscribe('list', list._id);
```

Publicaciones | publish-composite

```
Meteor.publishComposite('Todos.admin.inList', function(listId) {
  new SimpleSchema({
    listId: {type: String}
  }).validate({ listId });

  const userId = this.userId;
  return {
    find() {
      return Meteor.users.find({userId, admin: true});
    },
    children: [{
      find() {
        // We don't need to worry about the list.userId changing this time
        return [
          Lists.find(listId),
          Todos.find({listId})
        ];
      }
    }]
  };
});
```

Subscripciones

- Se realizan en el lado del cliente
- Utilizan **Meteor.subscribe()**
- Retorna un **“subscription handle”** con la propiedad llamada **.ready()**

```
const handle = Meteor.subscribe('lists.public');
```

- Existe también la propiedad **.stop()**
  - No es requerido en contextos reactivos
    - **Autorun**
    - **This.subscribe** (Blaze)

Subscripciones en UI components

Reactive Componet

```
Template.Lists_show_page.onCreated(function() {
  this.getListId = () => FlowRouter.getParam('_id');

  this.autorun(() => {
    this.subscribe('todos.inList', this.getListId());
  });
});
```

subscriptionsReady()

Subscripciones | paginación

Servidor

```
const MAX_TODOS = 1000;

Meteor.publish('todos.inList', function(listId, limit) {
  new SimpleSchema({
    listId: { type: String },
    limit: { type: Number }
  }).validate({ listId, limit });

  const options = {
    sort: {createdAt: -1},
    limit: Math.min(limit, MAX_TODOS)
  };

  // ...
});
```

Cliente

```
Template.Lists_show_page.onCreated(function() {
  this.getListId = () => FlowRouter.getParam('_id');


  this.autorun(() => {
    this.subscribe('todos.inList',
      this.getListId(), this.state.get('requestedTodos'));
  });
});
```

Subscripciones | paginación

Número de total de ítems (tmeasday:publish-counts**)**

```
Meteor.publish('Lists.todoCount', function({ listId }) {
  new SimpleSchema({
    listId: {type: String}
  }).validate({ listId });

  Counts.publish(this, `Lists.todoCount.${listId}`, Todos.find({listId}));
});
```

Cliente

```
Counts.get(`Lists.todoCount.${listId}`)
```

Conclusiones

- Buena información en el sitio de meteor
- La implementación del protocolo es sencilla
- Es importante seguir buenas prácticas y reglas definidas para dar seguridad a la aplicación.

Referencias

- https://www.meteor.com/
- https://guide.meteor.com/collections.html (collections)
- https://guide.meteor.com/methods.html
- https://guide.meteor.com/data-loading.html
- https://github.com/meteor/guide/blob/master/content/data-loading.md
- https://guide.meteor.com/accounts.html#userid-ddp