[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 0 - Getting Started](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Getting Started](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Blockchain 101](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Blockchain History](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Public and Managed Blockchains](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Consensus in Distributed Networks](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Cryptography](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Self-Assessment Quiz](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Go Introduction - First Steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Go Basics](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Go Interfaces](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Control Structures in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Arrays and Slices in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Standard Packages in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Concurrency in Go](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Docker Introduction](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Introduction to the Interchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[The Interchain Ecosystem](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Getting ATOM and Staking It](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[A Blockchain App Architecture](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Accounts](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Transactions](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Messages](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Modules](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Protobuf](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Multistore and Keepers](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[BaseApp](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Queries](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Events](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Context](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Testing](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Relaying with IBC](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Interchain Security](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Bridges](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 1 Quiz](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 2 - First Steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[First Steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Setup Your Work Environment](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Run a Node, API, and CLI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Ignite CLI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Store Object](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create and Save a Game Properly](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Add a Way to Make a Move](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Emit Game Information](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Record the Game Winner](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 2 Exercise](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[What is IBC?](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC Token Transfer](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC Tooling](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[What is CosmJS?](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Your First CosmJS Actions](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Compose Complex Transactions](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Learn to Integrate Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Put Your Games in Order](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Auto-Expiring Games](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Let Players Set a Wager](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Handle wager payments](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Integration tests](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Incentivize Players](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Help Find a Correct Move](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Understand IBC Denoms](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Go Relayer](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Hermes Relayer](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[CosmJS Advanced](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create Custom Objects](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create Custom Messages](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Get an External GUI](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Backend Script for Game Indexing](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC Deep Dive](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[IBC Application Developer Introduction](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Create a Leaderboard Chain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Run in Production](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Prepare the Software to Run](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Prepare a Validator and Keys](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Prepare Where the Node Starts](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Prepare and Do Migrations](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Simulate Production in Docker](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Tally Player Info After Production](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Simulate a Migration in Docker](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Final Exam](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[What's Next?](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

[Continue Your Interchain Journey](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html)

Docs Version Switcher

On this page

[Server idea](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#server-idea)

[Barebones server](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#barebones-server)

[Data types](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#data-types)

[Empty indexer module](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#empty-indexer-module)

[Files for execution](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#files-for-execution)

[Quick test](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#quick-test)

[Add CosmJS StargateClient](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#add-cosmjs-stargateclient)

[Handle blocks](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-blocks)

[Handle a transaction](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-a-transaction)

[Handle events](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-events)

[Handle one event](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-event)

[Handle one create event](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-create-event)

[Handle one play event](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-play-event)

[Test time](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time)

[Prepare for EndBlock](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#prepare-for-endblock)

[Handle one block's EndBlock](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-block-s-endblock)

[Test time of forfeit](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time-of-forfeit)

[Patch a game](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#patch-a-game)

[Test time of patch](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time-of-patch)

[Conclusion](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#conclusion)

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#backend-script-for-game-indexing) **Backend Script for Game Indexing**



Make sure you have all you need before proceeding:

* You understand the concepts of [CosmJS](https://ida.interchain.io/tutorials/7-cosmjs/1-cosmjs-intro.html).
* You have the checkers CosmJS codebase up to the integrated GUI. If not, follow the [previous steps](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html) or go ahead and clone and checkout [this branch (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/tree/gui) to get the version needed for this tutorial.

This exercise assumes that:

1. You are running your [checkers blockchain](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/2-cosmjs-messages.html#prepare-your-checkers-chain) with:



Copy

$ docker network create checkers-net

$ docker run --rm -it \

-p 26657:26657 \

--name checkers \

--network checkers-net \

--detach \

checkersd\_i:standalone start

$ sleep 10

$ docker run --rm -it \

-p 4500:4500 \

--name cosmos-faucet \

--network checkers-net \

--detach \

cosmos-faucet\_i:0.28.11 start http://checkers:26657

$ sleep 20

1. You have the following in .env:

**Local**

**Docker**



Copy

RPC\_URL="http://localhost:26657"

FAUCET\_URL="http://localhost:4500"

When you run npm on your local computer.

Copy

RPC\_URL="http://checkers:26657"

FAUCET\_URL="http://cosmos-faucet:4500"

When you run npm in a Docker container.

Now that your blockchain is complete, you can think about additional data and services that would add value without increasing cost or complexity on-chain.

For example, how do you list all of a player's games? Currently this information is not easily available. You can find the players of a given game, but not the games of a given player. Indexing this on-chain would add storage and computation costs.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#server-idea) Server idea

To implement this functionality, build a Web 2.0 server to do the indexing. The server:

1. Listens to updates from the checkers chain:
   1. On a game creation event, adds the game ID under each player.
   2. On a game deletion event, removes it.
2. When asked about its status, returns the latest block height up to which it has indexed players.
3. When asked about a given player, returns the list of game IDs for this player.
4. When a game ID is submitted, patches its information about that game ID on a best effort basis (to palliate potential de-synchronization).

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#barebones-server) Barebones server

As a fast and simple Web 2.0 solution, navigate to the [checkers CosmJS repository (opens new window)↗](https://github.com/cosmos/academy-checkers-ui) for the checkers blockchain, in the existing gui branch, and perform the following steps:

1. Create a sub-directory of the src folder (e.g. server).
2. Use the express Node.js module to create an HTTP REST API.
3. Use a local db.json as a *database*. This is obviously primitive and not thread-safe. In a production setting use a proper database.
4. Poll the blockchain at regular intervals. As part of an advanced topic, you can use WebSockets.

**Local**

**Docker**



Copy

$ npm install express@4.18.1 --save-exact

$ npm install @types/express@4.17.13 --save-dev --save-exact

Copy

$ docker run --rm \

-v $(pwd):/client \

-w /client \

node:18.7-slim \

npm install express@4.18.1 --save-exact

$ docker run --rm \

-v $(pwd):/client \

-w /client \

node:18.7-slim \

npm install @types/express@4.17.13 --save-dev --save-exact

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#data-types) Data types

To keep the code type-safe, define the types of your db.json in types.ts:



Copy

export interface LatestBlockStatus {

height: number

}

export interface DbStatus {

block: LatestBlockStatus

}

export interface PlayerInfo {

gameIds: string[]

}

export interface PlayersInfo {

[playerAddress: string]: PlayerInfo

}

export interface GameInfo {

redAddress: string

blackAddress: string

deleted: boolean

}

export interface GamesInfo {

[gameId: string]: GameInfo

}

export interface DbType {

status: DbStatus

players: PlayersInfo

games: GamesInfo

}

src /

server /

types.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/types.ts" \l "L1-L31" \t "_blank)

Not only does this keep information about players, it also keeps a copy of games. This gets around a current limitation of CosmJS, where you cannot get information about a game that has just been erased from the latest state. In practice you would need to query the game at an earlier block height, but this functionality is not yet available with CosmJS. Note that nodes may prune the old state, especially if they migrate, which may unpredictably impact any query at an earlier block height.

****

**Deleting data from blockchains**

State storage is the most expensive resource a blockchain application uses. Wasteful use of storage burdens nodes with unnecessarily large storage requirements.

From an engineering perspective, it is important to separate strict protocol needs from overall requirements. If no state-changing, on-chain logic relies on a stored value, then by definition that value's usefulness is limited to reporting. Such a value is a candidate for deletion from the blockchain state. Reporting is important, but consider more efficient methods of addressing it.

An off-chain indexer - think block explorers - can record historic facts before they are purged. This will support queries about details that are no longer present on the chain. This can also support queries about details that are still present on the chain, but at a much cheaper cost per query than querying blockchain nodes.

Preventing duplication of unique identifiers is a common requirement. The rule can be enforced even if the details are purged, as long as you keep a list of used keys or observed hashes that must not be used again. For example, keep IDs for games that used to exist while deleting the details about the moves and results that have no further relevance to the protocol because the games are "over". If needed, you can keep the complete move-by-move history of the game in cheaper, off-chain storage. This is possible if you designed your chain with proper reporting mechanisms, for instance via events.

In most cases, the most elegant code proceeds on the basis that referential integrity is guaranteed, avoiding messy exceptions like orphaned keys. Internal referential integrity is entirely the responsibility of the application developer, so consider using techniques like cascade deletions or preventing deletion if a record is referenced elsewhere.

Generally, it is a good idea to use deletes to prevent the state from simply growing forever. A simple rule of thumb is to delete everything you can, but no more.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#empty-indexer-module) Empty indexer module

Define a barebones server without any Cosmos elements in an indexer.ts. This is not CosmJS related, so start from something else if you prefer.



Copy

import { writeFile } from "fs/promises"

import { Server } from "http"

import express, { Express, Request, Response } from "express"

import { DbType } from "./types"

export const createIndexer = async () => {

const port = "3001"

const dbFile = `${\_\_dirname}/db.json`

const db: DbType = require(dbFile)

const pollIntervalMs = 5\_000 // 5 seconds

let timer: NodeJS.Timer | undefined

const app: Express = express()

app.get("/", (req: Request, res: Response) => {

res.send({

error: "Not implemented",

})

})

app.get("/status", (req: Request, res: Response) => {

res.json({

block: {

height: db.status.block.height,

},

})

})

app.get("/players/:playerAddress", (req: Request, res: Response) => {

res.json({

gameCount: db.players[req.params.playerAddress]?.gameIds?.length ?? 0,

gameIds: db.players[req.params.playerAddress]?.gameIds ?? [],

})

})

app.get("/players/:playerAddress/gameIds", (req: Request, res: Response) => {

res.json(db.players[req.params.playerAddress]?.gameIds ?? [])

})

app.patch("/games/:gameId", (req: Request, res: Response) => {

res.json({

result: "Not implemented",

})

})

const saveDb = async () => {

await writeFile(dbFile, JSON.stringify(db, null, 4))

}

const init = async () => {

setTimeout(poll, 1)

}

const poll = async () => {

console.log(new Date(Date.now()).toISOString(), "TODO poll")

timer = setTimeout(poll, pollIntervalMs)

}

process.on("SIGINT", () => {

if (timer) clearTimeout(timer)

saveDb()

.then(() => {

console.log(`${dbFile} saved`)

})

.catch(console.error)

.finally(() => {

server.close(() => {

console.log("server closed")

process.exit(0)

})

})

})

const server: Server = app.listen(port, () => {

init()

.catch(console.error)

.then(() => {

console.log(`\nserver started at http://localhost:${port}`)

})

})

}

Expand



src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \t "_blank)



Note:

1. The timer is set at the end of the previous poll, in case indexing takes longer than the interval.
2. The *database* is, for now, purely in memory as it runs and is saved on exit by catching the interruption signal.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#files-for-execution) Files for execution

Prepare these files around the indexer to run it in the terminal:

1. Create an index.ts that describes how to run the indexer:



Copy

require("./indexer").createIndexer().then(console.log).catch(console.error)

export {}

src /

server /

index.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/index.ts" \t "_blank)

The export {} prevents Visual Studio Code from complaining.

1. Add a specific tsconfig.json file if necessary:



Copy

{

"extends": "../../tsconfig.json",

"compilerOptions": {

"target": "ESNext",

"isolatedModules": false,

"module": "CommonJS"

}

}

src /

server /

tsconfig.json

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/tsconfig.json" \t "_blank)

1. In package.json, add a run target:



Copy

"scripts": {

...

+ "indexer-dev": "npx ts-node src/server/index.ts"

}

package.json

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/package.json" \l "L13" \t "_blank)

1. Add your *database*, db.json by making a copy of the sample:



Copy

{

"status": {

"block": {

"height": 0

}

},

"players": {},

"games": {}

}

src /

server /

db.json.sample

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/db.json.sample" \t "_blank)

You cannot ask for block at height 0, so the indexer first asks for the next block at 1.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#quick-test) Quick test

Check that the indexer works:

**Local**

**Docker**



Copy

$ npm run indexer-dev

Copy

$ docker run --rm -it \

-v $(pwd):/client -w /client \

-p 3001:3001 \

--network checkers-net \

--env RPC\_URL="http://checkers:26657" \

node:18.7-slim \

npm run indexer-dev

It should print:



Copy

> checkers-server@1.0.0 dev

> npx ts-node index.ts

server started at http://localhost:3001

Now, in another terminal, test the endpoints. Omit the | jq beautifier if it is not installed on your system:

**status**

**player info**

**player games**

**game update**



Copy

$ curl localhost:3001/status | jq

It should return:



Copy

{

"block": {

"height": 0

}

}

Copy

$ curl localhost:3001/players/cosmos123 | jq

It should return:

Copy

{

"gameCount": 0,

"gameIds": []

}

Copy

$ curl localhost:3001/players/cosmos123/gameIds | jq

It should return:

Copy

[]

Copy

$ curl -X PATCH localhost:3001/games/445 | jq

It should return:

Copy

{

"result": "Not implemented"

}

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#add-cosmjs-stargateclient) Add CosmJS StargateClient

You need to create a client to connect to your checkers blockchain. The client only needs read-only functionality because this server does not submit transactions. Your repository already contains useful elements:

* The CheckersStargateClient.
* An [.env (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/.env) file.

Add the following to indexer.ts:

1. The declarations:



Copy

import { config } from "dotenv"

import { CheckersStargateClient } from "../checkers\_stargateclient"

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L5" \t "_blank)

The pickup of RPC\_URL:



Copy

config()

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L14" \t "_blank)

The client in the indexer:



Copy

let client: CheckersStargateClient

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L22" \t "_blank)

1. The modified init:



Copy

const init = async() => {

client = await CheckersStargateClient.connect(process.env.RPC\_URL!)

console.log("Connected to chain-id:", await client.getChainId())

setTimeout(poll, 1)

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L64-L68" \t "_blank)

1. The modified poll:



Copy

const poll = async() => {

const currentHeight = await client.getHeight()

console.log(

new Date(Date.now()).toISOString(),

"Current heights:",

db.status.block.height,

"<=",

currentHeight,

)

timer = setTimeout(poll, pollIntervalMs)

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L70-L85" \t "_blank)

If you have not done it yet, start your checkers chain as described at the beginning of this section.

Relaunch the indexer:

**Local**

**Docker**



Copy

$ npm run indexer-dev

Copy

$ docker run --rm -it \

-v $(pwd):/client -w /client \

-p 3001:3001 \

--network checkers-net \

--env RPC\_URL="http://checkers:26657" \

node:18.7-slim \

npm run indexer-dev

Note how the container is started inside checkers-net alongside the checkers blockchain.

You should see the current height rising:



Copy

Connected to chain-id: checkers-1

server started at http://localhost:3001

2022-04-20T17:46:29.962Z Current heights: 0 <= 1353

2022-04-20T17:46:34.968Z Current heights: 0 <= 1357

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-blocks) Handle blocks

To index games, take each block and listen for the following relevant events:

1. A transaction with a new-game-created event.
2. An EndBlock with a game-forfeited event.

Start by getting each block from your last saved state. Update poll:



Copy

const poll = async () => {

const currentHeight = await client.getHeight()

if (db.status.block.height <= currentHeight - 100)

console.log(`Catching up ${db.status.block.height}..${currentHeight}`)

while (db.status.block.height < currentHeight) {

const processing = db.status.block.height + 1

process.stdout.cursorTo(0)

// Get the block

const block: Block = await client.getBlock(processing)

process.stdout.write(`Handling block: ${processing} with ${block.txs.length} txs`)

// Function yet to be declared

await handleBlock(block)

db.status.block.height = processing

}

await saveDb()

timer = setTimeout(poll, pollIntervalMs)

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L70-L85" \t "_blank)

This needs a new import:



Copy

import { Block } from "@cosmjs/stargate"

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L3" \t "_blank)

The indexer:

* Declares a new function handleBlock. Create one and put console.log(block) inside to explore what this object is and consider what actions you would take with it.
* Saves the db after a poll, so *you* can watch it in real time.
* Uses process.stdout.write and process.stdout.cursorTo(0) so that the repetitive logging happens on a single line.

Observe the relevant content in handleBlock. It must:

1. Extract the events from transactions.
2. Extract the events from EndBlock.

If you examine block.txs directly you find transactions as they were posted. However, this does not reveal any execution results, such as if a transaction executed as expected or what game ID it used for the new game. To get this extra information:

1. Calculate txHash from the transaction.
2. Call await client.getTx(txHash), which returns an IndexedTx.

The handleBlock function can be:



Copy

const handleBlock = async (block: Block) => {

if (0 < block.txs.length) console.log("")

let txIndex = 0

while (txIndex < block.txs.length) {

const txHash: string = toHex(sha256(block.txs[txIndex])).toUpperCase()

const indexed: IndexedTx | null = await client.getTx(txHash)

if (!indexed) throw new Error(`Could not find indexed tx: ${txHash}`)

// Function yet to be declared

await handleTx(indexed)

txIndex++

}

// TODO handle EndBlock

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L87-L96" \t "_blank)

This needs new imports:



Copy

+ import { sha256 } from "@cosmjs/crypto"

+ import { toHex } from "@cosmjs/encoding"

- import { Block } from "@cosmjs/stargate"

+ import { Block, IndexedTx } from "@cosmjs/stargate"

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L1-L3" \t "_blank)



* while() { await } simplifies the syntax of awaiting multiple times sequentially.
* The hash is calculated this way as per [here (opens new window)↗](https://github.com/cosmos/cosmjs/blob/v0.28.11/packages%2Fstargate%2Fsrc%2Fstargateclient.ts#L77).
* console.log("") puts a new line (poll does a process.stdout.write which adds no line).
* The handleBlock function uses a new function, handleTx. Create one and put console.log(indexed) inside to explore what this object is and consider what actions you can take with it.
* The EndBlock part has not yet been incorporated. This is explained in **Prepare for EndBlock**.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-a-transaction) Handle a transaction

Define the handleTx function:



Copy

const handleTx = async (indexed: IndexedTx) => {

const rawLog: any = JSON.parse(indexed.rawLog)

const events: StringEvent[] = rawLog.flatMap((log: ABCIMessageLog) => log.events)

// Function yet to be declared

await handleEvents(events)

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L102-L106" \t "_blank)

This needs new imports:



Copy

import { ABCIMessageLog, StringEvent } from "cosmjs-types/cosmos/base/abci/v1beta1/abci"

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L4" \t "_blank)



* [.flatMap (opens new window)↗](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/flatMap) transforms an array of arrays into a flattened array.
* The handleTx function uses a new function, handleEvents. Create one and put console.log(events) in it to explore what this object is and consider what actions you can take with it.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-events) Handle events

Define the handleEvents function:



Copy

const handleEvents = async (events: StringEvent[]): Promise<void> => {

try {

let eventIndex = 0

while (eventIndex < events.length) {

// Function yet to be declared

await handleEvent(events[eventIndex])

eventIndex++

}

} catch (e) {

// Skipping if the handling failed. Most likely the transaction failed.

}

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L108-L118" \t "_blank)



* while() {} simplifies the syntax of awaiting multiple times sequentially.
* The handleEvents function only keeps events that emanate from the checkers module.
* It uses a new function, handleEvent. Create one and put console.log(event) inside to explore what this object is and consider what actions you can take with it.
* It skips in case of error, as the likely cause is that the transactions in fact failed. This is not good enough if the goal is to be absolutely accurate.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-event) Handle one event

Define handleEvent as follows:



Copy

const handleEvent = async (event: StringEvent): Promise<void> => {

if (event.type == "new-game-created") {

// Function yet to be declared

await handleEventCreate(event)

}

if (event.type == "move-played") {

// Function yet to be declared

await handleEventPlay(event)

}

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L120-L129" \t "_blank)



* [new-game-created (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/cosmjs-elements/x/checkers/types/keys.go#L31) and [move-played (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/cosmjs-elements/x/checkers/types/keys.go#L41) are constant values defined in your Go code. They are put as the [event's type (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/cosmjs-elements/x/checkers/keeper/msg_server_create_game.go#L50).
* handleEvent uses two new functions: handleEventCreate and handleEventPlay. Create them and put console.log(event) inside each to explore what these objects are and consider what actions you would take with them.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-create-event) Handle one create event

Now update your db with the information provided. First, define a convenience function in createIndexer:



Copy

const getAttributeValueByKey = (attributes: Attribute[], key: string): string | undefined => {

return attributes.find((attribute: Attribute) => attribute.key === key)?.value

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L135-L137" \t "_blank)

This needs a new import:



Copy

- import { ABCIMessageLog, StringEvent } from "cosmjs-types/cosmos/base/abci/v1beta1/abci"

+ import { ABCIMessageLog, Attribute, StringEvent } from "cosmjs-types/cosmos/base/abci/v1beta1/abci"

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L4" \t "_blank)

Now define handleEventCreate as:



Copy

const handleEventCreate = async (event: StringEvent): Promise<void> => {

const newId: string | undefined = getAttributeValueByKey(event.attributes, "game-index")

if (!newId) throw new Error(`Create event missing game-index`)

const blackAddress: string | undefined = getAttributeValueByKey(event.attributes, "black")

if (!blackAddress) throw new Error(`Create event missing black address`)

const redAddress: string | undefined = getAttributeValueByKey(event.attributes, "red")

if (!redAddress) throw new Error(`Create event missing red address`)

console.log(`New game: ${newId}, black: ${blackAddress}, red: ${redAddress}`)

const blackInfo: PlayerInfo = db.players[blackAddress] ?? {

gameIds: [],

}

const redInfo: PlayerInfo = db.players[redAddress] ?? {

gameIds: [],

}

if (blackInfo.gameIds.indexOf(newId) < 0) blackInfo.gameIds.push(newId)

if (redInfo.gameIds.indexOf(newId) < 0) redInfo.gameIds.push(newId)

db.players[blackAddress] = blackInfo

db.players[redAddress] = redInfo

db.games[newId] = {

redAddress: redAddress,

blackAddress: blackAddress,

deleted: false,

}

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L139-L162" \t "_blank)



* [game-index (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/cosmjs-elements/x/checkers/keeper/msg_server_create_game.go#L52), [black (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/cosmjs-elements/x/checkers/keeper/msg_server_create_game.go#L53), and [red (opens new window)↗](https://github.com/cosmos/b9-checkers-academy-draft/blob/cosmjs-elements/x/checkers/keeper/msg_server_create_game.go#L54) are constants from the Go code.
* You have implemented error handling.
* handleEventCreate is careful not to double-add a given game ID.
* It does not save db as this is under the purview of poll().

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-play-event) Handle one play event

Not all play events are equal. handleEventPlay is only interested when there is a winner. Until then, there is no need to take any action.



Copy

const handleEventPlay = async (event: StringEvent): Promise<void> => {

const playedId: string | undefined = getAttributeValueByKey(event.attributes, "game-index")

if (!playedId) throw new Error(`Play event missing game-index`)

const winner: string | undefined = getAttributeValueByKey(event.attributes, "winner")

if (!winner) throw new Error("Play event missing winner")

if (winner === "\*") return

const blackAddress: string | undefined = db.games[playedId]?.blackAddress

const redAddress: string | undefined = db.games[playedId]?.redAddress

console.log(`Win game: ${playedId}, black: ${blackAddress}, red: ${redAddress}, winner: ${winner}`)

const blackGames: string[] = db.players[blackAddress]?.gameIds ?? []

const redGames: string[] = db.players[redAddress]?.gameIds ?? []

const indexInBlack: number = blackGames.indexOf(playedId)

if (0 <= indexInBlack) blackGames.splice(indexInBlack, 1)

const indexInRed: number = redGames.indexOf(playedId)

if (0 <= indexInRed) redGames.splice(indexInRed, 1)

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L179-L194" \t "_blank)



* handleEventPlay returns quietly if there is no winner, because this means there is nothing to do.
* It keeps the game information in the db.
* It removes the id from both players' list of games.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time) Test time

You can now test what happens when a game is created and played on. Restart npm run indexer-dev locally or in Docker.

You can choose how to create and play games:

* Run the GUI prepared in the [previous section](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html) with npm start.
* Run checkersd command lines.
* Any other way available.

Via command lines, in another terminal:

**Create game**

**Play game**



Copy

$ docker exec -it checkers sh -c \

"checkersd tx checkers \

create-game \$ALICE cosmos1t88fkwurlnusf6agvptsnm33t40kr4hlq6h08s 0 stake \

--from alice --keyring-backend test \

--yes"

Alternatively, use a GUI if preferred. The indexer should log something like:



Copy

New game: 1, black: cosmos1am3fnp5dd6nndk5jyjq9mpqh3yvt2jmmdv83xn, red: cosmos1t88fkwurlnusf6agvptsnm33t40kr4hlq6h08s

It should update db.json to:



Copy

{

"status": {

"block": {

"height": 100

}

},

"players": {

"cosmos1am3fnp5dd6nndk5jyjq9mpqh3yvt2jmmdv83xn": {

"gameIds": [

"1"

]

},

"cosmos1t88fkwurlnusf6agvptsnm33t40kr4hlq6h08s": {

"gameIds": [

"1"

]

}

},

"games": {

"1": {

"redAddress": "cosmos1t88fkwurlnusf6agvptsnm33t40kr4hlq6h08s",

"blackAddress": "cosmos1am3fnp5dd6nndk5jyjq9mpqh3yvt2jmmdv83xn",

"deleted": false

}

}

}

Copy

$ docker exec -it checkers \

checkersd tx checkers \

play-move 1 1 2 2 3 \

--from alice --keyring-backend test \

--yes

In this case the indexer should not log anything.

Because performing moves from the command line is laborious, using the GUI is advisable.

What remains is handling the games that get removed or forfeited in EndBlock.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#prepare-for-endblock) Prepare for EndBlock

Nicely formatted EndBlock events are still missing from CosmJS, so these require a little extra work:

1. To get a block's EndBlock events, you need to ask for the block information from a CometBFT client. This client is a [private field (opens new window)↗](https://github.com/cosmos/cosmjs/blob/902f21b/packages%2Fstargate%2Fsrc%2Fstargateclient.ts#L140) of StargateClient.
2. The function to call is [blockResults (opens new window)↗](https://github.com/cosmos/cosmjs/blob/5ee3f82/packages/tendermint-rpc/src/tendermint34/tendermint34client.ts#L88).
3. It returns a [BlockResultsResponse (opens new window)↗](https://github.com/cosmos/cosmjs/blob/ca969f2/packages/tendermint-rpc/src/tendermint34/responses.ts#L55), of which endBlockEvents: Event is of interest.
4. This [Event (opens new window)↗](https://github.com/cosmos/cosmjs/blob/ca969f2/packages/tendermint-rpc/src/tendermint34/responses.ts#L182) type has attributes: Attribute[] of interest.
5. The [Attribute (opens new window)↗](https://github.com/cosmos/cosmjs/blob/ca969f2/packages/tendermint-rpc/src/tendermint34/responses.ts#L177-L180) type is coded as Uint8Array.

With this information, you can do the necessary actions:

1. To handle the conversion of CometBFT Events into StringEvents, create a helper in a new src/server/events.ts:



Copy

import { fromUtf8 } from "@cosmjs/encoding"

import { Attribute as TendermintAttribute, Event } from "@cosmjs/tendermint-rpc"

import { Attribute, StringEvent } from "cosmjs-types/cosmos/base/abci/v1beta1/abci"

export const convertTendermintEvents = (events: readonly Event[]): StringEvent[] => {

return events.map(

(event: Event): StringEvent => ({

type: event.type,

attributes: event.attributes.map(

(attribute: TendermintAttribute): Attribute => ({

key: fromUtf8(attribute.key),

value: fromUtf8(attribute.value),

}),

),

}),

)

}

src /

server /

events.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/events.ts" \l "L1-L17" \t "_blank)

1. To handle the call to blockResults, you need access to a CometBFT client. One option is to make a copy of the private CometBFT client. You can do this only on construction, so create a child class of CheckersStargateClient to do that. It is recommended to keep it close by indexer.ts. In a new indexer\_stargateclient.ts:



Copy

import { StargateClientOptions } from "@cosmjs/stargate"

import { BlockResultsResponse, Tendermint34Client } from "@cosmjs/tendermint-rpc"

import { StringEvent } from "cosmjs-types/cosmos/base/abci/v1beta1/abci"

import { convertTendermintEvents } from "./events"

import { CheckersStargateClient } from "../checkers\_stargateclient"

export class IndexerStargateClient extends CheckersStargateClient {

private readonly myTmClient: Tendermint34Client

public static async connect(

endpoint: string,

options: StargateClientOptions = {},

): Promise<IndexerStargateClient> {

const tmClient = await Tendermint34Client.connect(endpoint)

return new IndexerStargateClient(tmClient, options)

}

protected constructor(tmClient: Tendermint34Client, options: StargateClientOptions) {

super(tmClient, options)

this.myTmClient = tmClient

}

public async getEndBlockEvents(height: number): Promise<StringEvent[]> {

const results: BlockResultsResponse = await this.myTmClient.blockResults(height)

return convertTendermintEvents(results.endBlockEvents)

}

}

src /

server /

indexer\_stargateclient.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer_stargateclient.ts" \t "_blank)

Now swap out CheckersStargateClient with IndexerStargateClient:



Copy

import { IndexerStargateClient } from "./indexer\_stargateclient"

export const createIndexer = async () => {

...

let client: IndexerStargateClient

...

const init = async () => {

client = await IndexerStargateClient.connect(process.env.RPC\_URL!)

...

}

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L22-L65" \t "_blank)

With this in place, go back to handleBlock and work on the remaining TODO.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-block-s-endblock) Handle one block's EndBlock

Go to the function and update it:



Copy

const handleBlock = async (block: Block) => {

...

const events: StringEvent[] = await client.getEndBlockEvents(block.header.height)

if (0 < events.length) console.log("")

await handleEvents(events)

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L97-L99" \t "_blank)

The events that you have converted are compatible with those emanating from transactions, so you can just pass them on. You still need to update handleEvent so that it acts on the new event type:



Copy

const handleEvent = async (event: StringEvent): Promise<void> => {

...

+ if (event.type == "game-forfeited") {

+ // Function yet to be declared

+ await handleEventForfeit(event)

+ }

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L130-L132" \t "_blank)

To achieve this, add a new function:



Copy

const handleEventForfeit = async (event: StringEvent): Promise<void> => {

const forfeitedId: string | undefined = getAttributeValueByKey(event.attributes, "game-index")

if (!forfeitedId) throw new Error(`Forfeit event missing forfeitedId`)

const winner: string | undefined = getAttributeValueByKey(event.attributes, "winner")

const blackAddress: string | undefined = db.games[forfeitedId]?.blackAddress

const redAddress: string | undefined = db.games[forfeitedId]?.redAddress

console.log(

`Forfeit game: ${forfeitedId}, black: ${blackAddress}, red: ${redAddress}, winner: ${winner}`,

)

const blackGames: string[] = db.players[blackAddress]?.gameIds ?? []

const redGames: string[] = db.players[redAddress]?.gameIds ?? []

const indexInBlack: number = blackGames.indexOf(forfeitedId)

if (0 <= indexInBlack) blackGames.splice(indexInBlack, 1)

const indexInRed: number = redGames.indexOf(forfeitedId)

if (0 <= indexInRed) redGames.splice(indexInRed, 1)

if (db.games[forfeitedId]) db.games[forfeitedId].deleted = true

}

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L196-L212" \t "_blank)



Again there is a lot of error handling. handleEvent only soft-deletes the game, although it removes it from the list of games for the players.

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time-of-forfeit) Test time of forfeit

Run the previous tests again. Create a game and see how the deletion event is picked up:



Copy

Forfeit game: 1, black: cosmos1am3fnp5dd6nndk5jyjq9mpqh3yvt2jmmdv83xn, red: cosmos1t88fkwurlnusf6agvptsnm33t40kr4hlq6h08s, winner: \*



In the standalone checkers in Docker, the deadline is unfortunately set at 24 hours, so feedback is not exactly coming fast. At this state of the exercise, if you want to test the expiry quickly, you will have to run Ignite CLI and adjust the MaxTurnDuration as described [here](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/4-game-forfeit.html#interact-via-the-cli).

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#patch-a-game) Patch a game

In the actions that the Express server exposes, app.patch still remains to be implemented. This allows a user to inform the server that its database is no longer synchronized, and that it should look at a specific game. It is a matter of data re-synchronization:

1. If the game can be found in the blockchain state, update the indexer's database accordingly:
   1. If there is a winner, then the game should be removed from its players' lists of games.
   2. If there is no winner, then the game should be added to its players' lists of games.
2. If the game cannot be found in the blockchain state, but is present in the indexer's database, then the game should be removed from the lists of games of its players, and marked as soft-deleted. This shows the usefulness of keeping *old* games.
3. If the game cannot be found either in the blockchain state nor in the indexer's database, then it is better not to do anything. To remove it from all players' lists of games is potentially expensive. This could expose the server to a DoS attack.

Code the following:



Copy

const patchGame = async (gameId: string): Promise<boolean> => {

const game: StoredGame | undefined = await client.checkersQueryClient?.checkers.getStoredGame(gameId)

const cachedGame: GameInfo | undefined = db.games[gameId]

if (!game && cachedGame) {

console.log(

`Patch game: deleted, ${gameId}, black: ${cachedGame.blackAddress}, red: ${cachedGame.redAddress}`,

)

const blackGames: string[] = db.players[cachedGame.blackAddress]?.gameIds ?? []

const redGames: string[] = db.players[cachedGame.redAddress]?.gameIds ?? []

const indexInBlack: number = blackGames.indexOf(gameId)

if (0 <= indexInBlack) blackGames.splice(indexInBlack, 1)

const indexInRed: number = redGames.indexOf(gameId)

if (0 <= indexInRed) redGames.splice(indexInRed, 1)

cachedGame.deleted = true

return true

} else if (!game) {

// No information to work from.

// If we try to remove it from all players, it is very expensive and we are at risk of a DoS attack.

console.log(`Patch game: not found, ${gameId}`)

return false

} else if (game.winner !== "\*") {

const blackGames: string[] = db.players[game.black]?.gameIds ?? []

const redGames: string[] = db.players[game.red]?.gameIds ?? []

console.log(

`Patch game: ended, ${gameId}, black: ${game.black}, red: ${game.red}, winner: ${game.winner}`,

)

const indexInBlack: number = blackGames.indexOf(gameId)

if (0 <= indexInBlack) blackGames.splice(indexInBlack, 1)

const indexInRed: number = redGames.indexOf(gameId)

if (0 <= indexInRed) redGames.splice(indexInRed, 1)

return true

} else {

const blackInfo: PlayerInfo = db.players[game.black] ?? {

gameIds: [],

}

const redInfo: PlayerInfo = db.players[game.red] ?? {

gameIds: [],

}

console.log(`Patch game: new, ${gameId}, black: ${game.black}, red: ${game.red}`)

if (blackInfo.gameIds.indexOf(gameId) < 0) blackInfo.gameIds.push(gameId)

if (redInfo.gameIds.indexOf(gameId) < 0) redInfo.gameIds.push(gameId)

db.players[game.black] = blackInfo

db.players[game.red] = redInfo

db.games[gameId] = {

redAddress: game.red,

blackAddress: game.black,

deleted: false,

}

return true

}

}

Expand



src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L214-L264" \t "_blank)

There are some issues to be aware of:

1. JavaScript is not thread-safe, so you could cause two opposite actions: one coming from the polling and the other from a patch submission, or even from two concurrent patch submissions. To reduce this risk the *database* is not saved to disk in this function, but instead relies on the polling to save it at the next run.
2. Assuming that *there is no such game when you cannot find it* can result in deleting data that is simply taking time to appear on your blockchain node.

Next, you need to call patchGame from the app.patch callback:



Copy

app.patch("/games/:gameId", async (req: Request, res: Response) => {

const found = await patchGame(req.params.gameId)

if (!found) res.status(404)

else {

res.json({

result: "Thank you",

})

}

})

src /

server /

indexer.ts

[View source→](https://github.com/cosmos/academy-checkers-ui/blob/server-indexing/src/server/indexer.ts" \l "L50-L58" \t "_blank)

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time-of-patch) Test time of patch

To simulate a case where the game is in the blockchain state but not the indexer's:

1. Stop your indexer.
2. Create a game and check at what block it is included (for example, at index 3 and block 1001).
3. Update your indexer's db.json to pretend that it already indexed the game's block by setting:



Copy

"status": {

"block": {

"height": 1001

}

}

1. Restart the indexer.
2. From another terminal, make a call to it:



Copy

$ curl -X PATCH localhost:3001/games/3 | jq

It should return:



Copy

{

"result": "Thank you"

}

And the indexer should log something like:



Copy

Patch game: new, 3, black: cosmos1am3fnp5dd6nndk5jyjq9mpqh3yvt2jmmdv83xn, red: cosmos1t88fkwurlnusf6agvptsnm33t40kr4hlq6h08s

Develop your own ways to test the other scenarios.

If you started the chain in Docker, when you are done you can stop the containers with:



Copy

$ docker stop cosmos-faucet checkers

$ docker network rm checkers-net

[#Copy link](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#conclusion) Conclusion

You have created a small server that:

* Polls the blockchain to get events about created, won, and forfeited games.
* Maintains a database with information indexed in real time.
* Offers this information as a Web service.
* Accepts requests for patches.

These are examples of server-side scripts, which can improve user experience.

You can find the complete code [here (opens new window)↗](https://github.com/cosmos/academy-checkers-ui/tree/server-indexing).

previous

[](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)

**[Integrate CosmJS and Keplr](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/4-cosmjs-gui.html)**

up next

**[IBC Deep Dive](https://ida.interchain.io/ida-course/LPs/week-6/)**

[[](https://ida.interchain.io/ida-course/LPs/week-6/)](https://ida.interchain.io/ida-course/LPs/week-6/)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[Server idea](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#server-idea)

[Barebones server](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#barebones-server)

[Data types](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#data-types)

[Empty indexer module](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#empty-indexer-module)

[Files for execution](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#files-for-execution)

[Quick test](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#quick-test)

[Add CosmJS StargateClient](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#add-cosmjs-stargateclient)

[Handle blocks](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-blocks)

[Handle a transaction](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-a-transaction)

[Handle events](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-events)

[Handle one event](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-event)

[Handle one create event](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-create-event)

[Handle one play event](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-play-event)

[Test time](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time)

[Prepare for EndBlock](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#prepare-for-endblock)

[Handle one block's EndBlock](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#handle-one-block-s-endblock)

[Test time of forfeit](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time-of-forfeit)

[Patch a game](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#patch-a-game)

[Test time of patch](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#test-time-of-patch)

[Conclusion](https://ida.interchain.io/hands-on-exercise/3-cosmjs-adv/5-server-side.html#conclusion)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)