

# @push.rocks/smartdata

An advanced library for NoSQL data organization and manipulation using TypeScript with support for MongoDB, data validation, collections, and custom data types.

- [readme.md for @push.rocks/smartdata](#)
- [changelog.md for @push.rocks/smartdata](#)

# readme.md for @push.rocks/smartdata

[npm version](#)

**The ultimate TypeScript-first MongoDB ODM** — type-safe decorators, real-time change streams, Lucene-powered search, distributed leader election, and cursor streaming. Built for modern applications that demand performance, correctness, and developer experience.

## Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit [community.foss.global/](https://community.foss.global/). This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a [code.foss.global/](https://code.foss.global/) account to submit Pull Requests directly.

## ☐ Why SmartData?

- ☐ **100% Type-Safe** — TC39 Stage 3 decorators, generic filters, and compile-time query validation
- < **High Performance** — Connection pooling, cursor streaming, and automatic indexing
- ☐ **Real-time Ready** — MongoDB Change Streams with RxJS for reactive applications
- ☐ **Distributed Systems** — Built-in leader election and task coordination via `@push.rocks/taskbuffer`
- ☐ **Security First** — `$where` injection prevention, operator allow-listing, and input sanitization
- ☐ **Lucene Search** — Full-text, wildcard, boolean, and range queries out of the box
- ☐ **Great DX** — Intuitive API, IntelliSense that just works, and lifecycle hooks

## ☐ Installation

```
pnpm add @push.rocks/smartdata
```

# ☐ Requirements

- **Node.js**  $\geq 20.x$
- **Deno**  $\geq 2.0$  (for Deno projects)
- **MongoDB**  $\geq 5.0$
- **TypeScript**  $\geq 5.2$  (for TC39 decorator support)

“ **Note:** SmartData uses TC39 Stage 3 decorators (the standard). Make sure `experimentalDecorators` is **not** set in your `tsconfig.json`. Bun is not currently supported as it doesn't implement TC39 decorators yet.

# ☐ Quick Start

## 1☐ Connect to Your Database

```
import { SmartdataDb } from '@push.rocks/smartdata';

const db = new SmartdataDb({
  mongoDbUrl: 'mongodb://localhost:27017/myapp',
  mongoDbName: 'myapp',
  mongoDbUser: 'username',
  mongoDbPass: 'password',
});

await db.init();
console.log(db.status); // 'connected'
```

## 2☐ Define Your Data Models

```
import {
  SmartDataDbDoc,
```

```

Collection,
unI,
svDb,
index,
searchable,
} from '@push.rocks/smartdata';

@Collection(() => db)
class User extends SmartDataDbDoc<User, User> {
  @unI()
  public id!: string;

  @svDb()
  @searchable()
  public username!: string;

  @svDb()
  @searchable()
  @index({ unique: false })
  public email!: string;

  @svDb()
  public status!: 'active' | 'inactive' | 'pending';

  @svDb()
  public tags!: string[];

  @svDb()
  public createdAt: Date = new Date();

  constructor(username: string, email: string) {
    super();
    this.username = username;
    this.email = email;
  }
}

```

## 3 CRUD Operations

```
// Create
const user = new User('johndoe', 'john@example.com');
user.status = 'active';
user.tags = ['developer', 'typescript'];
await user.save();

// Read – fully type-safe filters
const foundUser = await User.getInstance({ username: 'johndoe' });
const activeUsers = await User.getInstances({ status: 'active' });

// Update
foundUser.email = 'newemail@example.com';
await foundUser.save();

// Delete
await foundUser.delete();
```

## Features

### Type-Safe Query Filters

SmartData provides a rich, type-safe filtering system supporting all MongoDB operators with full IntelliSense:

```
// Comparison operators
const adults = await User.getInstances({
  age: { $gte: 18, $lt: 65 },
});

// Array operators
const experts = await User.getInstances({
  tags: { $all: ['typescript', 'mongodb'] },
  skills: { $size: 5 },
});

// Logical operators
const complex = await Order.getInstances({
```

```

$and: [
  { status: 'active' },
  { $or: [{ priority: 'high' }, { value: { $gte: 1000 } }] },
],
});

// Deep nested object queries
const users = await User.getInstances({
  profile: {
    settings: {
      notifications: { email: true },
    },
  },
});

// Dot notation
const sameUsers = await User.getInstances({
  'profile.settings.notifications.email': true,
});

// Regex patterns
const gmailUsers = await User.getInstances({
  email: { $regex: '@gmail\\.com$', $options: 'i' },
});

```

**Security:** The `$where` operator is automatically blocked to prevent NoSQL injection. Unknown operators trigger warnings.

## 📦 Lucene-Powered Search

Mark fields with `@searchable()` to enable a built-in search engine with automatic compound text indexing:

```

@Collection(() => db)
class Product extends SmartDataDbDoc<Product, Product> {
  @unI() public id!: string;
  @svDb() @searchable() public name!: string;
  @svDb() @searchable() public description!: string;
  @svDb() @searchable() public category!: string;
}

```

```
@svDb() public price!: number;
}

// Simple text search across all @searchable fields
const results = await Product.search('laptop');

// Field-specific search
const electronics = await Product.search('category:Electronics');

// Wildcard
const matches = await Product.search('Mac*');

// Boolean operators (AND, OR, NOT)
const query = await Product.search('laptop AND NOT gaming');

// Phrase search
const exact = await Product.search('"MacBook Pro"');

// Range queries
const midRange = await Product.search('price:[100 TO 500]');

// Combined with MongoDB filters and post-fetch validation
const affordable = await Product.search('laptop', {
  filter: { price: { $lte: 1500 } },
  validate: async (p) => p.price > 0,
});
```

## ☐☐ Real-Time Change Streams

Watch for database changes with RxJS subjects and EventEmitter support:

```
const watcher = await User.watch(
  { status: 'active' },
  {
    fullDocument: 'updateLookup',
    bufferTimeMs: 100, // optional: buffer changes via RxJS
  },
);
```

```
// RxJS subscription
watcher.changeSubject.subscribe((user) => {
  console.log('User changed:', user);
});

// Or EventEmitter style
watcher.on('change', (user) => {
  console.log('User changed:', user);
});

// Clean up
await watcher.close();
```

## Cursor Streaming

Process large datasets without memory pressure:

```
const cursor = await User.getCursor(
  { status: 'active' },
  {
    modifier: (c) => c.sort({ createdAt: -1 }).limit(10000),
  },
);

// Iterate one-by-one
await cursor.forEach(async (user) => {
  await processUser(user);
});

// Or collect into an array
const users = await cursor.toArray();

// Always close when done
await cursor.close();
```

## Transactions

Ensure atomic consistency across multiple operations:

```

const session = db.startSession();

try {
  await session.withTransaction(async () => {
    const sender = await User.getInstance({ id: 'user-1' }, { session });
    sender.balance -= 100;
    await sender.save({ session });

    const receiver = await User.getInstance({ id: 'user-2' }, { session });
    receiver.balance += 100;
    await receiver.save({ session });
  });
} finally {
  await session.endSession();
}

```

## ☐☐ EasyStore — Type-Safe Key-Value Storage

Built on top of SmartData collections, EasyStore provides simple key-value persistence:

```

interface AppConfig {
  apiKey: string;
  features: { darkMode: boolean; notifications: boolean };
  limits: { maxUsers: number };
}

const config = await db.createEasyStore<AppConfig>('app-config');

// Write
await config.writeKey('features', { darkMode: true, notifications: false });

// Read
const features = await config.readKey('features');
// TypeScript knows: features.darkMode is boolean ☐

// Read all

```

```
const all = await config.readAll();

// Write multiple keys
await config.writeAll({ apiKey: 'new-key', limits: { maxUsers: 500 } });

// Delete a key
await config.deleteKey('features');

// Wipe the store
await config.wipe();
```

## ☐☐ Distributed Coordination

Built-in leader election using MongoDB for coordination, integrating with `@push.rocks/taskbuffer`:

```
import { SmartdataDistributedCoordinator } from '@push.rocks/smartdata';

const coordinator = new SmartdataDistributedCoordinator(db);

// Start coordination – automatic heartbeat and leader election
await coordinator.start();

// Fire distributed task requests
const result = await coordinator.fireDistributedTaskRequest({
  submitterId: 'instance-1',
  requestResponseId: 'unique-id',
  taskName: 'process-payments',
  taskVersion: '1.0.0',
  taskExecutionTime: Date.now(),
  taskExecutionTimeout: 30000,
  taskExecutionParallel: 1,
  status: 'requesting',
});

// Graceful shutdown with leadership handoff
await coordinator.stop();
```

## ☐☐ Custom Serialization

Transform data on its way in and out of MongoDB:

```
@Collection(() => db)
class Doc extends SmartDataDbDoc<Doc, Doc> {
  @svDb({
    serialize: (set) => Array.from(set),
    deserialize: (arr) => new Set(arr),
  })
  public tags!: Set<string>;

  @svDb({
    serialize: (date) => date?.toISOString(),
    deserialize: (str) => (str ? new Date(str) : null),
  })
  public scheduledAt!: Date | null;
}
```

## ☐ Lifecycle Hooks

Add custom logic before and after save/delete:

```
@Collection(() => db)
class Order extends SmartDataDbDoc<Order, Order> {
  @unI() public id!: string;
  @svDb() public items!: Array<{ product: string; quantity: number; price: number }>;
  @svDb() public totalAmount!: number;

  async beforeSave() {
    this.totalAmount = this.items.reduce((s, i) => s + i.price * i.quantity, 0);
  }

  async afterSave() {
    await notificationService.orderUpdated(this.id);
  }

  async beforeDelete() {
    if (this.totalAmount > 0) throw new Error('Cannot delete non-zero orders');
  }
}
```

```
async afterDelete() {
  await cache.delete(`order:${this.id}`);
}
}
```

## ☐☐ Indexing

```
@Collection(() => db)
class HighPerformanceDoc extends SmartDataDbDoc<HighPerformanceDoc, HighPerformanceDoc> {
  @unI()
  public id!: string; // Unique index

  @index()
  public userId!: string; // Regular index

  @index({ sparse: true })
  public deletedAt?: Date; // Sparse index – only indexes docs where field exists

  @index({ expireAfterSeconds: 86400 })
  public sessionToken!: string; // TTL index – auto-expires after 24h
}
```

## ☐☐ Connection Options

```
const db = new SmartdataDb({
  mongoDbUrl: 'mongodb://localhost:27017',
  mongoDbName: 'myapp',
  mongoDbUser: 'admin',
  mongoDbPass: 's3cret',

  // Connection pool tuning (all optional)
  maxPoolSize: 100,           // Max connections (default: 100)
  maxIdleTimeMS: 300000,     // Close idle connections after 5min (default)
  serverSelectionTimeoutMS: 30000, // Timeout for server selection
  socketTimeoutMS: 30000,    // Socket timeout to prevent hung operations
});
```

# Decorators Reference

Decorator	Target	Description
<code>@Collection(dbGetter)</code>	Class	Binds a document class to a MongoDB collection
<code>@managed(managerGetter?)</code>	Class	Like <code>@Collection</code> but controlled by a manager instance
<code>@unI()</code>	Field	Marks as unique index + saveable
<code>@svDb(options?)</code>	Field	Marks field as saveable, with optional <code>serialize</code> / <code>deserialize</code>
<code>@index(options?)</code>	Field	Creates a regular MongoDB index
<code>@searchable()</code>	Field	Enables Lucene-style text search on this field
<code>@globalSvDb()</code>	Field	Marks field as globally saveable across all doc types

# API Reference

## Core Classes

Class	Description
<code>SmartdataDb</code>	Database connection, session management, EasyStore factory
<code>SmartDataDbDoc&lt;T, TImpl&gt;</code>	Base class for all document models
<code>SmartdataCollection&lt;T&gt;</code>	Underlying collection manager (usually accessed indirectly)
<code>SmartdataDbCursor&lt;T&gt;</code>	Cursor for streaming large result sets
<code>SmartdataDbWatcher&lt;T&gt;</code>	Change stream watcher with RxJS + EventEmitter
<code>SmartdataDistributedCoordinator</code>	Leader election and distributed task coordination
<code>EasyStore&lt;T&gt;</code>	Type-safe key-value store backed by a collection

## Key Static Methods on `SmartDataDbDoc`

Method	Description
<code>getInstances(filter, opts?)</code>	Find multiple documents
<code>getInstance(filter, opts?)</code>	Find a single document (or <code>null</code> )
<code>getCursor(filter, opts?)</code>	Get a streaming cursor
<code>getCount(filter?)</code>	Count matching documents
<code>watch(filter, opts?)</code>	Watch for real-time changes
<code>search(query, opts?)</code>	Lucene-style full-text search
<code>forEach(filter, fn)</code>	Iterate all matches with a callback
<code>getNewId(length?)</code>	Generate a class-prefixed unique ID
<code>createSearchFilter(luceneQuery)</code>	Convert Lucene query to MongoDB filter
<code>getSearchableFields()</code>	List all <code>@searchable()</code> fields

## Key Instance Methods on `SmartDataDbDoc`

Method	Description
<code>save(opts?)</code>	Insert or update the document
<code>delete(opts?)</code>	Delete the document
<code>updateFromDb()</code>	Refresh fields from the database
<code>saveDeep(savedMap?)</code>	Recursively save referenced documents
<code>createSavableObject()</code>	Serialize to a plain object for persistence
<code>createIdentifiableObject()</code>	Extract unique index fields for filtering

## License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the [LICENSE](#) file.

**Please note:** The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

## Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

## Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at [hello@task.vc](mailto:hello@task.vc).

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

# changelog.md for @push.rocks/smartdata

## 2026-03-26 - 7.1.3 - fix(deps)

bump development dependencies for tooling and Node types

- update @git.zone/tsrun from ^2.0.1 to ^2.0.2
- update @git.zone/tstest from ^3.5.1 to ^3.6.0
- update @types/node from ^22.15.2 to ^25.5.0

## 2026-03-24 - 7.1.2 - fix(docs)

refresh project guidance for TC39 decorators, build configuration, and dependency compatibility

- streamlines readme hints to focus on current decorator patterns and runtime support
- adds compatibility notes for the updated build toolchain and dependency APIs
- includes the project license file in the repository

## 2026-03-24 - 7.1.1 - fix(build)

update build and test tooling configuration, migrate project config to .smartconfig.json, and align TypeScript typings

- Switch the build script to tsbuild tsfolders and upgrade core build/test dependencies including @git.zone/tsbuild, @git.zone/tstest, and @git.zone/tsrun.
- Replace npmextra.json with .smartconfig.json and update package packaging to include the new config file.
- Update test files to import tapbundle from @git.zone/tstest/tapbundle and remove the standalone @push.rocks/tapbundle dependency.
- Adjust TypeScript configuration and source typings for stricter compatibility, including node types and definite assignment/nullability fixes.

- Fix Gitea workflow repository URLs for code.foss.global and expand .gitignore for generated Rust and local tooling directories.

## 2026-02-26 - 7.1.0 - feat(config)

normalize npmextra.json to namespaced keys and add CI/release configuration

- Replaced legacy keys (npmdocker, npmci, gitzone, tsdoc) with namespaced package keys (@git.zone/cli, @git.zone/tsdoc, @git.zone/tsdocker, @ship.zone/szci).
- Moved tsdoc legal text under @git.zone/tsdoc.
- Added release configuration with registries (https://verdaccio.lossless.digital and https://registry.npmjs.org) and accessLevel public under @git.zone/cli.
- Added @git.zone/tsdocker CI/docker settings and @ship.zone/szci npm registry/tooling settings.
- Removed old top-level entries to consolidate tooling configuration under scoped keys.

## 2026-02-26 - 7.0.16 - fix(mongodb)

set default socketTimeoutMS to 30000ms in MongoClient options to prevent hung operations from holding connections

- Adds socketTimeoutMS: 30000 to MongoClient clientOptions in ts/classes.db.ts
- Helps prevent hung operations from indefinitely holding connections by enforcing a 30s socket timeout
- Non-breaking change (defaults only)

## 2025-12-01 - 7.0.15 - fix(classes.doc)

Avoid emitting instance fields for collection and manager to preserve decorator-defined prototype getters

- ts/classes.doc.ts: changed instance properties `collection` and `manager` to `declare` so TypeScript does not emit them as own properties — prevents ES2022 class fields from shadowing prototype getters created by `@Collection` and `@managed` decorators.
- readme.hints.md: added documentation explaining the ES2022 class fields issue and recommending use of `declare` for type-only instance properties; marks the fix as v7.0.15.

# 2025-11-28 - 7.0.14 - fix(classes.collection)

Centralize TC39 decorator metadata initialization and use context.metadata in class decorators

- Add initializeDecoratorMetadata helper to initialize prototype and constructor properties from TC39 decorator metadata
- Refactor Collection and managed decorators to call initializeDecoratorMetadata with context.metadata
- Remove direct reliance on constructor[Symbol.metadata] in class decorators to avoid read-only assignment issues
- Ensure consistent initialization of saveableProperties, globalSaveableProperties, uniqueIndexes, regularIndexes, searchableFields and \_svDbOptions

# 2025-11-28 - 7.0.13 - fix(classes.doc)

Remove noisy debug logging from decorators and serialization logic

- Removed debug logger calls from globalSvDb decorator initialization
- Removed debug logger calls from svDb decorator initialization and svDb options handling
- Removed debug logger calls from unI and index decorator initializers
- Removed debug logging in createSavableObject to reduce console noise; no functional changes

# 2025-11-28 - 7.0.12 - fix(collection)

Ensure TC39 decorator metadata is initialized on both original and decorated constructors/prototypes and add debug logging

- Initialize metadata-driven prototype properties (globalSaveableProperties, saveableProperties, uniqueIndexes, regularIndexes) on both the decorated class prototype and the original constructor prototype to avoid closure/compatibility issues

- Initialize searchableFields on both the decorated constructor and the original constructor so text-index creation and searches see the fields correctly
- Forward and initialize \_svDbOptions from decorator metadata onto the original constructor to preserve custom serialization options
- Add debug logging in the Collection decorator and in createSavableObject to surface metadata and saveable-property counts for easier troubleshooting

## 2025-11-28 - 7.0.9 - fix(classes.collection)

Fix closure bug in Collection decorator by defining collection getter on original constructor and prototype

- Define the collection getter on the original constructor so class-level references (e.g. `User.collection`) resolve to the decorated collection instead of the original constructor's closure value.
- Also define the getter on the original constructor's prototype to ensure instance access works consistently across runtimes (Deno/Node).

## 2025-11-28 - 7.0.8 - fix(classes.collection)

Fix closure issue in managed decorator so Class.collection/instance.collection resolve correctly

- Resolve closure bug in the managed() decorator where class methods referencing Class.collection (or instance.collection) could receive the original constructor's captured value and thus the wrong collection/manager.
- Define dynamic getters on the original constructor and its prototype that compute the collection from the proper manager/db at access time (supports direct manager objects, delayed manager factory functions, and fallback to defaultManager).
- Getters are defined as non-enumerable and configurable to preserve compatibility with existing consumers.

## 2025-11-28 - 7.0.7 - fix(decorators)

Fix decorator metadata initialization and Lucene query transformation

- Ensure TC39 decorator metadata is used to initialize prototype properties so decorators work reliably across runtimes (context.metadata / Symbol.metadata shim imported early).
- Field and class decorators now populate and consume metadata for saveable properties, indexes and searchable fields so prototype initialization happens before instance creation.
- Fix Lucene -> MongoDB transformer to produce correct \$or/\$and/\$not structures and improve wildcard/fuzzy/range handling for search queries.
- Improve collection initialization to auto-create compound text indexes from searchableFields and ensure index creation is idempotent.

## 2025-11-28 - 7.0.6 - fix(classes.collection)

Guard against missing collection before attaching document constructor in Collection decorator

- Added a truthy check for `coll` before setting `(coll as any).docCtor` in the Collection decorator (`ts/classes.collection.ts`).
- Prevents a potential `TypeError` when `collectionFactory.getCollection` returns `null/undefined` during decorator initialization.

## 2025-11-28 - 7.0.5 - fix(package)

Add package exports entry and remove legacy main/typings fields

- Added an "exports" entry in package.json mapping "." to `./dist_ts/index.js` to declare the package's ESM entrypoint.
- Removed legacy "main" and "typings" fields from package.json.
- Improves Node/module resolution and modern bundler compatibility by using the package exports field.

## 2025-11-28 - 7.0.4 - fix(decorators)

Add Symbol.metadata polyfill and import it at entry to ensure decorator metadata is available

- Add `ts/shim.ts`: defines `Symbol.metadata` when missing (polyfill for TC39 Stage 3 decorator metadata).
- Import `'./shim.js'` at the very top of `ts/index.ts` so the polyfill runs before any decorator code or exports are evaluated.

- Prevents runtime errors when decorators rely on Symbol.metadata and improves compatibility across runtimes/environments.

## 2025-11-28 - 7.0.3 - fix(build)

Bump devDependency @git.zone/tsbuild to ^3.1.2

- Updated @git.zone/tsbuild in devDependencies from ^3.1.1 to ^3.1.2

## 2025-11-28 - 7.0.2 - fix(collectionfactory)

Simplify CollectionFactory.getCollection: remove unnecessary IIFE and instantiate collection only when dbArg is SmartdataDb

- Remove redundant IIFE wrapper in getCollection for improved readability
- Only create and cache a SmartdataCollection when dbArg is an instance of SmartdataDb
- Avoid assigning undefined to the collections map by guarding instantiation and returning existing collection

## 2025-11-27 - 7.0.1 - fix(build)

Update build tooling and TypeScript compilation target

- Bump devDependency @git.zone/tsbuild from ^3.1.0 to ^3.1.1.
- Update tsconfig.json compiler target from ES2022 to ES2024 (affects emitted JS language level).

## 2025-11-27 - 7.0.0 - BREAKING CHANGE(mongodb)

Upgrade dependencies: bump mongodb to ^7.0.0 and @git.zone/tstest to ^3.1.3

- Bump 'mongodb' dependency from ^6.20.0 to ^7.0.0 — major version upgrade; may introduce breaking API changes and require code updates or verification against the new driver.
- Update devDependency '@git.zone/tstest' from ^2.8.1 to ^3.1.3 — test tooling updated.

## 2025-11-17 - 6.0.0 - BREAKING CHANGE(decorators)

Migrate to TC39 Stage 3 decorators and refactor decorator metadata handling; update class initialization, lucene adapter fixes and docs

- Switch all decorators to TC39 Stage 3 signatures and metadata usage (use context.metadata and context.addInitializer) — affects svDb, globalSvDb, searchable, unl, index, Collection and managed.
- Refactor Collection/managed decorators to read and initialize prototype/constructor properties from context.metadata to ensure prototype properties are available before instance creation (ts/classes.collection.ts).
- Improve search implementation: add a Lucene parser and transformer with safer MongoDB query generation, wildcard/fuzzy handling and properly structured boolean operators (ts/classes.lucene.adapter.ts).
- Search integration updated to use the new adapter and handle advanced Lucene syntax and edge cases more robustly.
- Bump dev tooling versions: @git.zone/tsbuild -> ^3.1.0 and @git.zone/tsrun -> ^2.0.0.
- Documentation: update README and add readme.hints.md describing the TC39 decorator migration, minimum TypeScript (>=5.2) and Deno notes; tests adjusted accordingly.
- Clean up project memory/config files related to the previous decorator approach and Deno configuration adjustments.

## 2025-11-17 - 5.16.7 - fix(classes.collection)

Improve Deno and TypeScript compatibility: Collection decorator \_svDbOptions forwarding and config cleanup

- Collection decorator: capture original constructor and forward \_svDbOptions to ensure property decorator options (serialize/deserialize) remain accessible in Deno environments.

- Collection decorator: keep instance getter defined on prototype for Deno compatibility (no behavior change, clarifies forwarding logic).
- Build/config: removed experimentalDecorators and useDefineForClassFields from deno.json and tsconfig.json to avoid Deno/TS build issues and rely on default compilation settings.

## 2025-11-17 - 5.16.6 - fix(classes)

Add Deno compatibility, prototype-safe decorators and safe collection accessor; bump a few deps

- Add deno.json to enable experimentalDecorators and target ES2022/DOM for Deno builds.
- Introduce getCollectionSafe() on SmartDataDbDoc and use it for save/update/delete/findOne to avoid runtime errors when instance 'collection' is not present.
- Change several instance properties (globalSaveableProperties, uniqueIndexes, regularIndexes, saveableProperties) to 'declare' so decorator-set prototype properties are not shadowed (Deno compatibility).
- Enhance @Collection decorator: capture original constructor/prototype for Deno, define prototype getter for collection on decorated class, attach docCtor for searchableFields, and forward \_svDbOptions to the original constructor to preserve serializer metadata.
- Improve text/search index handling by relying on docCtor.searchableFields and guarding text index creation.
- Bump dependencies/devDependencies: @push.rocks/smartmongo -> ^2.0.14, @git.zone/tsbuild -> ^2.7.1, @git.zone/tstest -> ^2.8.1.
- These are non-breaking runtime compatibility and developer-experience fixes; intended as a patch release.

## 2025-11-16 - 5.16.5 - fix(watcher)

Update dependencies, tooling and watcher import; add .serena cache ignore

- Bump runtime dependencies: @push.rocks/smartlog 3.1.8 → 3.1.10, @push.rocks/smartstring 4.0.15 → 4.1.0, @push.rocks/taskbuffer 3.1.7 → 3.4.0, @tsclass/tsclass 9.2.0 → 9.3.0, mongodb 6.18.0 → 6.20.0
- Bump devDependencies: @git.zone/tsbuild 2.6.7 → 2.6.8, @git.zone/tsrun 1.2.44 → 1.6.2, @git.zone/tstest 2.3.5 → 2.6.2
- Switch EventEmitter import to node:events in ts/classes/watcher.ts to use the namespaced Node import
- Add .serena/.gitignore to ignore /cache

# 2025-08-18 - 5.16.4 - fix(classes.doc (convertFilterForMongoDb))

Improve filter conversion: handle logical operators, merge operator objects, add nested filter tests and docs, and fix test script

- Fix package.json test script: remove stray dot in tctest --verbose argument to ensure tests run correctly
- Enhance convertFilterForMongoDb in ts/classes.doc.ts to properly handle logical operators (\$and, \$or, \$nor, \$not) and return them recursively
- Merge operator objects for the same field path (e.g. combining \$gte and \$lte) to avoid overwriting operator clauses when object and dot-notation are mixed
- Add validation/guards for operator argument types (e.g. \$in, \$nin, \$all must be arrays; \$size must be numeric) and preserve existing behavior blocking \$where for security
- Add comprehensive nested filter tests in test/test.filters.ts to cover deep nested object queries, \$elemMatch, array size, \$all, \$in on nested fields and more
- Expand README filtering section with detailed examples for basic filtering, deep nested filters, comparison operators, array operations, logical and element operators, and advanced patterns

# 2025-08-18 - 5.16.3 - fix(docs)

Add local Claude settings and remove outdated codex.md

- Added .claude/settings.local.json to store local Claude/assistant permissions and configuration.
- Removed codex.md (project overview) — documentation file deleted.
- No runtime/library code changes; documentation/configuration-only update, bump patch version.

# 2025-08-18 - 5.16.2 - fix(readme)

Update README: clarify examples, expand search/cursor/docs and add local Claude settings

- Refined README wording and structure: clearer Quick Start, improved examples and developer-focused phrasing
- Expanded documentation for search, cursors, change streams, distributed coordination, transactions and EasyStore with more concrete code examples
- Adjusted code examples to show safer defaults (ID generation, status/tags, connection pooling) and improved best-practices guidance
- Added `.claude/settings.local.json` to provide local assistant/CI permission configuration

## 2025-08-12 - 5.16.1 - fix(core)

Improve error handling and logging; enhance search query sanitization; update dependency versions and documentation

- Replaced `console.log` and `console.warn` with structured `logger.log` calls throughout the core modules
- Enhanced database initialization with `try/catch` and proper URI credential encoding
- Improved search query conversion by disallowing dangerous operators (e.g. `$where`) and securely escaping regex patterns
- Bumped dependency versions (`smartlog`, `@tsclass/tsclass`, `mongodb`, etc.) in `package.json`
- Added detailed project memories including code style, project overview, and suggested commands for developers
- Updated README with improved instructions, feature highlights, and quick start sections

## 2025-04-25 - 5.16.0 - feat(watcher)

Enhance change stream watchers with buffering and `EventEmitter` support; update dependency versions

- Bumped `smartmongo` from `^2.0.11` to `^2.0.12` and `smartrx` from `^3.0.7` to `^3.0.10`
- Upgraded `@tsclass/tsclass` to `^9.0.0` and `mongodb` to `^6.16.0`
- Refactored the `watch` API to accept additional options (`bufferTimeMs`, `fullDocument`) for improved change stream handling
- Modified `SmartdataDbWatcher` to extend `EventEmitter` and support event notifications

## 2025-04-24 - 5.15.1 - fix(cursor)

Improve cursor usage documentation and refactor getCursor API to support native cursor modifiers

- Updated examples in readme.md to demonstrate manual iteration using cursor.next() and proper cursor closing.
- Refactored the getCursor method in classes.doc.ts to accept session and modifier options, consolidating cursor handling.
- Added new tests in test/test.cursor.ts to verify cursor operations, including limits, sorting, and skipping.

## 2025-04-24 - 5.15.0 - feat(svDb)

Enhance svDb decorator to support custom serialization and deserialization options

- Added an optional options parameter to the svDb decorator to accept serialize/deserialize functions
- Updated instance creation logic (updateFromDb) to apply custom deserialization if provided
- Updated createSavableObject to use custom serialization when available

## 2025-04-23 - 5.14.1 - fix(db operations)

Update transaction API to consistently pass optional session parameters across database operations

- Revised transaction support in readme to use startSession without await and showcased session usage in getInstance and save calls
- Updated methods in classes.collection.ts to accept an optional session parameter for findOne, getCursor, findAll, insert, update, delete, and getCount
- Enhanced SmartDataDbDoc save and delete methods to propagate session parameters
- Improved overall consistency of transactional APIs across the library

## 2025-04-23 - 5.14.0 - feat(doc)

Implement support for beforeSave, afterSave, beforeDelete, and afterDelete lifecycle hooks in document save and delete operations to allow custom logic execution during these critical moments.

- Calls beforeSave hook if defined before performing insert or update.
- Calls afterSave hook after a document is saved.
- Calls beforeDelete hook before deletion and afterDelete hook afterward.
- Ensures \_updatedAt timestamp is refreshed during save operations.

## 2025-04-22 - 5.13.1 - fix(search)

Improve search query parsing for implicit AND queries by preserving quoted substrings and better handling free terms, quoted phrases, and field:value tokens.

- Replace previous implicit AND logic with tokenization that preserves quoted substrings
- Support both free term and field:value tokens with wildcards inside quotes
- Ensure errors are thrown for non-searchable fields in field-specific queries

## 2025-04-22 - 5.13.0 - feat(search)

Improve search query handling and update documentation

- Added 'codex.md' providing a high-level project overview and detailed search API documentation.
- Enhanced search parsing in SmartDataDbDoc to support combined free-term and quoted field phrase queries.
- Introduced a new fallback branch in the search method to handle free term with quoted field input.
- Updated tests in test/test.search.ts to cover new combined query scenarios and ensure robust behavior.

## 2025-04-22 - 5.12.2 - fix(search)

Fix handling of quoted wildcard patterns in field-specific search queries and add tests for location-based wildcard phrase searches

- Strip surrounding quotes from wildcard patterns in field queries to correctly transform them to regex
- Introduce new tests in test/test.search.ts to validate exact quoted and unquoted wildcard searches on a location field

# 2025-04-22 - 5.12.1 - fix(search)

Improve implicit AND logic for mixed free term and field queries in search and enhance wildcard field handling.

- Updated regex for field:value parsing to capture full value with wildcards.
- Added explicit handling for free terms by converting to regex across searchable fields.
- Improved error messaging for attempts to search non-searchable fields.
- Extended tests to cover combined free term and wildcard field searches, including error cases.

# 2025-04-22 - 5.12.0 - feat(doc/search)

Enhance search functionality with filter and validate options for advanced query control

- Added 'filter' option to merge additional MongoDB query constraints in search
- Introduced 'validate' hook to post-process and filter fetched documents
- Refactored underlying execQuery function to support additional search options
- Updated tests to cover new search scenarios and fallback mechanisms

# 2025-04-22 - 5.11.4 - fix(search)

Implement implicit AND logic for mixed simple term and field:value queries in search

- Added a new branch to detect and handle search queries that mix field:value pairs with plain terms without explicit operators
- Builds an implicit \$and filter when query parts contain colon(s) but lack explicit boolean operators or quotes
- Ensures proper parsing and improved robustness of search filters

# 2025-04-22 - 5.11.3 - fix(lucene adapter and search tests)

Improve range query parsing in Lucene adapter and expand search test coverage

- Added a new 'testSearch' script in package.json to run search tests.
- Introduced advanced search tests for range queries and combined field filters in test/search.advanced.ts.
- Enhanced robustness tests in test/search.ts for wildcard and empty query scenarios.
- Fixed token validation in the parseRange method of the Lucene adapter to ensure proper error handling.

## 2025-04-21 - 5.11.2 - fix(readme)

Update readme to clarify usage of searchable fields retrieval

- Replaced getSearchableFields('Product') with Product.getSearchableFields()
- Updated documentation to reference the static method Class.getSearchableFields()

## 2025-04-21 - 5.11.1 - fix(doc)

Refactor searchable fields API and improve collection registration.

- Removed the standalone getSearchableFields utility in favor of a static method on document classes.
- Updated tests to use the new static method (e.g., Product.getSearchableFields()).
- Ensured the Collection decorator attaches a docCtor property to correctly register searchable fields.
- Added try/catch in test cleanup to gracefully handle dropDatabase errors.

## 2025-04-21 - 5.11.0 -

## feat(ts/classes.lucene.adapter)

Expose luceneWildcardToRegex method to allow external usage and enhance regex transformation capabilities.

- Changed luceneWildcardToRegex from private to public in ts/classes.lucene.adapter.ts.

# 2025-04-21 - 5.10.0 - feat(search)

Improve search functionality: update documentation, refine Lucene query transformation, and add advanced search tests

- Updated readme.md with detailed Lucene-style search examples and use cases
- Enhanced LuceneToMongoTransformer to properly handle wildcard conversion and regex escaping
- Improved search query parsing in SmartDataDbDoc for field-specific, multi-term, and advanced Lucene syntax
- Added new advanced search tests covering boolean operators, grouping, quoted phrases, and wildcard queries

# 2025-04-18 - 5.9.2 - fix(documentation)

Update search API documentation to replace deprecated searchWithLucene examples with the unified search(query) API and clarify its behavior.

- Replaced 'searchWithLucene' examples with 'search(query)' in the README.
- Updated explanation to detail field-specific exact match, partial word regex search, multi-word literal matching, and handling of empty queries.
- Clarified guidelines for creating MongoDB text indexes on searchable fields for optimized search performance.

# 2025-04-18 - 5.9.1 - fix(search)

Refactor search tests to use unified search API and update text index type casting

- Replaced all calls from searchWithLucene with search in test/search tests
- Updated text index specification in the collection class to use proper type casting

# 2025-04-18 - 5.9.0 - feat(collections/search)

Improve text index creation and search fallback mechanisms in collections and document search methods

- Auto-create a compound text index on all searchable fields in SmartdataCollection with a one-time flag to prevent duplicate index creation.
- Refine the search method in SmartDataDbDoc to support exact field matches and safe regex fallback for non-Lucene queries.

## 2025-04-17 - 5.8.4 - fix(core)

Update commit metadata with no functional code changes

- Commit info and documentation refreshed
- No code or test changes detected in the diff

## 2025-04-17 - 5.8.3 - fix(readme)

Improve readme documentation on data models and connection management

- Clarify that data models use @Collection, @unl, @svDb, @index, and @searchable decorators
- Document that ObjectId and Buffer fields are stored as BSON types natively without extra decorators
- Update connection management section to use 'db.close()' instead of 'db.disconnect()'
- Revise license section to reference the MIT License without including additional legal details

## 2025-04-14 - 5.8.2 - fix(classes.doc.ts)

Ensure collection initialization before creating a cursor in getCursorExtended

- Added 'await collection.init()' to guarantee that the MongoDB collection is initialized before using the cursor
- Prevents potential runtime errors when accessing collection.mongodbCollection

# 2025-04-14 - 5.8.1 - fix(cursor, doc)

Add explicit return types and casts to SmartdataDbCursor methods and update getCursorExtended signature in SmartDataDbDoc.

- Specify Promise<T> as return type for next() in SmartdataDbCursor and cast return value to T.
- Specify Promise<T[]> as return type for toArray() in SmartdataDbCursor and cast return value to T[].
- Update getCursorExtended to return Promise<SmartdataDbCursor<T>> for clearer type safety.

# 2025-04-14 - 5.8.0 - feat(cursor)

Add toArray method to SmartdataDbCursor to convert raw MongoDB documents into initialized class instances

- Introduced asynchronous toArray method in SmartdataDbCursor which retrieves all documents from the MongoDB cursor
- Maps each native document to a SmartDataDbDoc instance using createInstanceFromMongoDbNativeDoc for consistent API usage

# 2025-04-14 - 5.7.0 - feat(SmartDataDbDoc)

Add extended cursor method getCursorExtended for flexible cursor modifications

- Introduces getCursorExtended in classes.doc.ts to allow modifier functions for MongoDB cursors
- Wraps the modified cursor with SmartdataDbCursor for improved API consistency
- Enhances querying capabilities by enabling customized cursor transformations

# 2025-04-07 - 5.6.0 - feat(indexing)

Add support for regular index creation in documents and collections

- Implement new index decorator in classes.doc.ts to mark properties with regular indexing options
- Update SmartdataCollection to create regular indexes if defined on a document during insert
- Enhance document structure to store and utilize regular index configurations

## 2025-04-06 - 5.5.1 - fix(ci & formatting)

Minor fixes: update CI workflow image and npmci package references, adjust package.json and readme URLs, and apply consistent code formatting.

- Update image and repo URL in Gitea workflows from GitLab to code.foss.global
- Replace '@shipzone/npmci' with '@ship.zone/npmci' throughout CI scripts
- Adjust homepage and bugs URL in package.json and readme
- Apply trailing commas and consistent formatting in TypeScript source files
- Minor update to .gitignore custom section label

## 2025-04-06 - 5.5.0 - feat(search)

Enhance search functionality with robust Lucene query transformation and reliable fallback mechanisms

- Improve Lucene adapter to properly structure \$or queries for term, phrase, wildcard, and fuzzy search
- Implement and document a robust searchWithLucene method with fallback to in-memory filtering
- Update readme and tests with extensive examples for @searchable fields and Lucene-based queries

## 2025-04-06 - 5.4.0 - feat(core)

Refactor file structure and update dependency versions

- Renamed files and modules from 'smartdata.classes.' to 'classes.' and adjusted corresponding import paths.
- Updated dependency versions: '@push.rocks/smartmongo' to ^2.0.11, '@tsclass/tsclass' to ^8.2.0, and 'mongodb' to ^6.15.0.
- Renamed dev dependency packages from '@gitzone/...' to '@git.zone/...' and updated '@push.rocks/tapbundle' and '@types/node'.
- Fixed YAML workflow command: replaced 'pnpm install -g @gitzone/tsdoc' with 'pnpm install -g @git.zone/tsdoc'.
- Added package manager configuration and pnpm-workspace.yaml for built dependencies.

## 2025-03-10 - 5.3.0 - feat(docs)

Enhance documentation with updated installation instructions and comprehensive usage examples covering advanced features such as deep queries, automatic indexing, and distributed coordination.

- Added pnpm installation command
- Updated User model example to include ObjectId, Binary, and custom serialization
- Expanded CRUD operations examples with cursor methods and deep query support
- Enhanced sections on EasyStore, real-time data watching with RxJS integration, and managed collections
- Included detailed examples for transactions, deep object queries, and document lifecycle hooks

## 2025-02-03 - 5.2.12 - fix(documentation)

Remove license badge from README

- Removed the license badge from the README file, ensuring compliance with branding guidelines.

## 2025-02-03 - 5.2.11 - fix(documentation)

Updated project documentation for accuracy and added advanced feature details

- Added details for EasyStore, Distributed Coordination, and Real-time Data Watching features.
- Updated database connection setup instructions to include user authentication.
- Re-organized advanced usage section to showcase additional features separately.

## 2024-09-05 - 5.2.10 - fix(smartdata.classes.doc)

Fix issue with array handling in convertFilterForMongoDb function

- Corrected the logic to properly handle array filters in the convertFilterForMongoDb function to avoid incorrect assignments.

## 2024-09-05 - 5.2.9 - fix(smartdata.classes.doc)

Fixed issue with convertFilterForMongoDb to handle array operators.

- Updated the convertFilterForMongoDb function in smartdata.classes.doc.ts to properly handle array operators like \$in and \$all.

## 2024-09-05 - 5.2.8 - fix(smartdata.classes.doc)

Fix key handling in convertFilterForMongoDb function

- Fixed an issue in convertFilterForMongoDb that allowed keys with dots which could cause errors.

## 2024-09-05 - 5.2.7 - fix(core)

Fixed issue with handling filter keys containing dots in smartdata.classes.doc.ts

- Fixed an error in the `convertFilterForMongoDb` function which previously threw an error when keys contained dots.

## 2024-06-18 - 5.2.6 - Chore

### Maintenance Release

- Release version 5.2.6

## 2024-05-31 - 5.2.2 - Bug Fixes

### Fixes and Maintenance

- Fixed issue where `_createdAt` and `_updatedAt` registered `saveableProperties` for all document types

## 2024-04-15 - 5.1.2 - New Feature

### Enhancements and Bug Fixes

- Added static `.getCount({})` method to `SmartDataDbDoc`
- Changed fields `_createdAt` and `_updatedAt` to ISO format

## 2024-04-14 - 5.0.43 - New Feature

### New Feature Addition

- Added default `_createdAt` and `_updatedAt` fields, fixes #1

## 2024-03-30 - 5.0.41 - Bug Fixes

### Improvements and Fixes

- Improved `tsconfig.json` for ES Module use

# 2023-07-10 - 5.0.20 - Chore

## Organizational Changes

- Switched to new org scheme

# 2023-07-21 - 5.0.21 to 5.0.26 - Fixes

## Multiple Fix Releases

- Various core updates and bug fixes

# 2023-07-21 - 5.0.20 - Chore

## Organizational Changes

- Switch to the new org scheme

# 2023-06-25 - 5.0.14 to 5.0.19 - Fixes

## Multiple Fix Releases

- Various core updates and bug fixes

# 2022-05-17 - 5.0.0 - Major Update

## Breaking Changes

- Switched to ESM

# 2022-05-18 - 5.0.2 - Bug Fixes

## Bug Fixes

- The `watcher.changeSubject` now emits the correct type into observer functions

# 2022-05-17 - 5.0.1 - Chore

## Testing Improvements

- Tests now use `@pushrocks/smartmongo` backed by `wiredTiger`

# 2022-05-17 to 2022-11-08 - 5.0.8 to 5.0.10

## Multiple Fix Releases

- Various core updates and bug fixes

# 2021-11-12 - 4.0.17 to 4.0.20

## Multiple Fix Releases

- Various core updates and bug fixes

# 2021-09-17 - 4.0.10 to 4.0.16

## Multiple Fix Releases

- Various core updates and bug fixes

# 2021-06-09 - 4.0.1 to 4.0.9

Multiple Fix Releases

- Various core updates and bug fixes

## 2021-06-06 - 4.0.0 - Major Update

Major Release

- Maintenance and core updates

## 2021-05-17 - 3.1.56 - Chore

Maintenance Release

- Release version 3.1.56

## 2020-09-09 - 3.1.44 to 3.1.52

Multiple Fix Releases

- Various core updates and bug fixes

## 2020-06-12 - 3.1.26 to 3.1.28

Multiple Fix Releases

- Various core updates and bug fixes

## 2020-02-18 - 3.1.23 to 3.1.25

Multiple Fix Releases

- Various core updates and bug fixes

# 2019-09-11 - 3.1.20 to 3.1.22

Multiple Fix Releases

- Various core updates and bug fixes

# 2018-07-10 - 3.0.5 - New Feature

Added Feature

- Added custom unique indexes to `SmartdataDoc`

# 2018-07-08 - 3.0.1 - Chore

Dependencies Update

- Updated mongodb dependencies

# 2018-07-08 - 3.0.0 - Major Update

Refactor and Cleanup

- Cleaned project structure

# 2018-01-16 - 2.0.7 - Breaking Change

Big Changes

- Switched to `@pushrocks` scope and moved from `rethinkdb` to `mongodb`

# 2018-01-12 - 2.0.0 - Major Release

## Core Updates

- Updated CI configurations