

# Using Moor

---



**Simone Alessandria**

AUTHOR, TRAINER AND PROUD DEVELOPER

[www.softwarehouse.it](http://www.softwarehouse.it)



# Overview



About Moor

Using Moor in a Flutter Project

Tables

CRUD

Streams

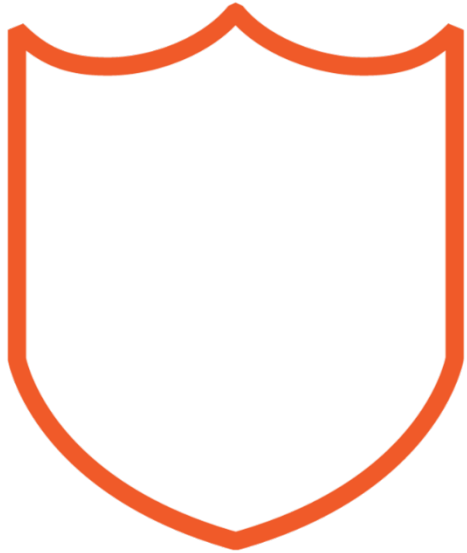


# Moor

Moor is a **reactive persistence library** for Dart and Flutter applications.



# Moor Features



Type Safety



Stream Queries



Data Validation



Less Code

```

part 'moor_database.g.dart';

class BlogPosts extends Table {

  IntColumn get id =>
    integer().autoIncrement().call();

  RealColumn get someFloat => real();

  TextColumn get name =>
    text().withLength(min: 1, max: 80);

  BlobColumn get image =>
    blob();

  DateTimeColumn get date =>
    dateTime().nullable();

  BoolColumn get isImportant =>
    boolean().withDefault(Constant(false))
    ();

}

```

- ◀ part statement
- ◀ **Table** class
- ◀ At the end of the Column definition you add a call() method
- ◀ You can also add () instead of .call()
- ◀ Define field properties: withLength, nullable, withDefault, autoIncrement...
- ◀ DateTime is saved as a Unix TimeStamp
- ◀ Booleans are saved as integers



```
Future<List<BlogPost>> getPosts() =>  
select(blogPosts).get();
```

```
Stream<List<BlogPost>> getPosts() =>  
select(blogPosts).watch();
```

```
Future<int> insertPost(BlogPost  
post) => into(blogPosts).insert(post);
```

```
Future<bool> updatePost(BlogPost  
post) =>  
update(blogPosts).replace(post);
```

```
Future<int> deletePost(BlogPost  
post) =>  
delete(blogPosts).delete(post);
```

◀ Returns a List of objects (**Future**)

◀ Returns a List of objects (**Stream**)

◀ Insert

◀ Update

◀ Delete



```
BlogPostsCompanion.insert({  
    this.id = const Value.absent(),  
    required String name,  
    this.content = const Value.absent(),  
    this.date = const Value.absent(),  
})
```

## Companion Objects

**Insert or update partial data**



# Summary



## Moor

- Create, Read, Update, Delete
- Future and Stream

## Code Generation

- part

## Provider

