

GopherCon Africa 2026 Workshop



Hi

ainsley.dev

godaily.dev

ainsleyclark.com

Domain-Driven Applications with sqlc and templ

Why?

Why?

Why?

10 mins

Welcome

20 mins

App Intro

50 mins

**Architecture &
DDD Practices**

40 mins

**Querying with
sqlc**

40 mins

Intro to templ

20 mins

Profiling

80 mins

Theming

10 mins

Wrapping Up

10 mins

Welcome

20 mins

App

20 mins

App Intro

50 mins

**Ar
DD**

50 mins

Architecture & DDD Practices

40 m

Qu
sq

40 mins

Querying with sqlc

40 mins

Intro to templ

20 m

Pro

20 mins

Profiling

80 mins

Th

80 mins

Theming

10 m

Wi

10 mins

Wrapping Up

10 mins

Welcome

20 mins

App Intro

50 mins

**Architecture &
DDD Practices**

40 mins

**Querying with
sqlc**

40 mins

Intro to templ

20 mins

Profiling

80 mins

Theming

10 mins

Wrapping Up

So welcome all...

So welcome all...

What you'll learn...

Setup

Questions

Follow along

Ok let's get setup

Ok let's get setup

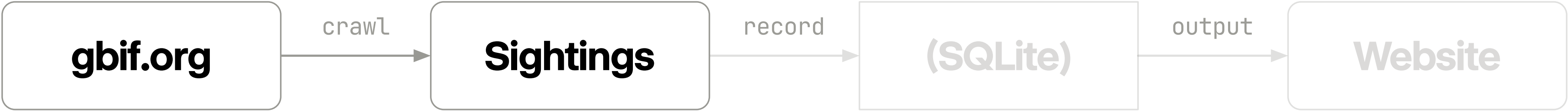
architectures

What you'll leave with

The app



The app



The app



The app



gbif.org

open-meteo.com

open-meteo.com

Give it a try

Have a play

Competitions

Best Social Post

Tag @Ainsley Clark, @JetBrains, @GopherCon Africa on LinkedIn

Picture of you or what you made.

Or even the features in GoLand (Air)

Best Website

Best design or function.

Built in your fork during the Theming block.

Pushed to GitHub by the end of the day.

Best Website

Best design or function.

Built in your fork during the Theming block.

Pushed to GitHub by the end of the day.

Architecture & DDD Practices

Architecture & DDD Practices

Five folders, one feature

What?

Why?

DDD is not a directory layout

DDD is not a directory layout

DDD is not a directory layout



Nothing

Flat

Wh

B

What it does

By layer

Ri

C

Rings

Onion

On

H

One per noun

Hexagonal

At

F

All the way down

Feature

Question one

**Can you tell what it
does?**

Question two

**Where does a new
feature go?**

Question three

**Where do the rules
live?**

Question one

**Can you tell what it
does?**

Question two

**What
features**

Question two

**Where does a new
feature go?**

Questi

**Where
live?**

at it

Question three

ew

**Where do the rules
live?**

Benefits

What's good

Weaknesses

What's bad

Improve it

**What would we
change**

Benefits

What's good

Weakne

Wha

Weaknesses

What's bad

Improv

**What
chan**

Improve it

**What would we
change**

The layout



The layout



The layout



The layout



The layout



01

ORGANISED BY NOTHING

Flat

1	01-flat/	
2	├── models.go	Species, Occurrence, GBIF's shapes
3	├── gbif.go	API client
4	├── ingest.go	Ingest data & seed
5	├── store.go	In-memory store
6	├── handler.go	GET /, and the display-name rule
7	├── sightings.html	HTML page
8	└── main.go	Wiring

1	01-flat/	
2	├─ models.go	Species, Occurrence, GBIF's shapes
3	├─ gbif.go	API client
4	├─ ingest.go	Ingest data & seed
5	├─ store.go	In-memory store
6	├─ handler.go	GET /, and the display-name rule
7	├─ sightings.html	HTML page
8	└─ main.go	Wiring

1	01-flat/	
2	├── models.go	Species, Occurrence, GBIF's shapes
3	├── gbif.go	API client
4	├── ingest.go	Ingest data & seed
5	├── store.go	In-memory store
6	├── handler.go	GET /, and the display-name rule
7	├── sightings.html	HTML page
8	└── main.go	Wiring

1	01-flat/	
2	├── models.go	Species, Occurrence, GBIF's shapes
3	├── gbif.go	API client
4	├── ingest.go	Ingest data & seed
5	├── store.go	In-memory store
6	├── handler.go	GET /, and the display-name rule
7	├── sightings.html	HTML page
8	└── main.go	Wiring

Three questions

Can you tell what it does?

Where does a new feature go?

Where do the rules live?

Flat: Pros

Speed

One edit, not ten

Shape

**Nothing to
navigate**

Speed

One edit, not ten

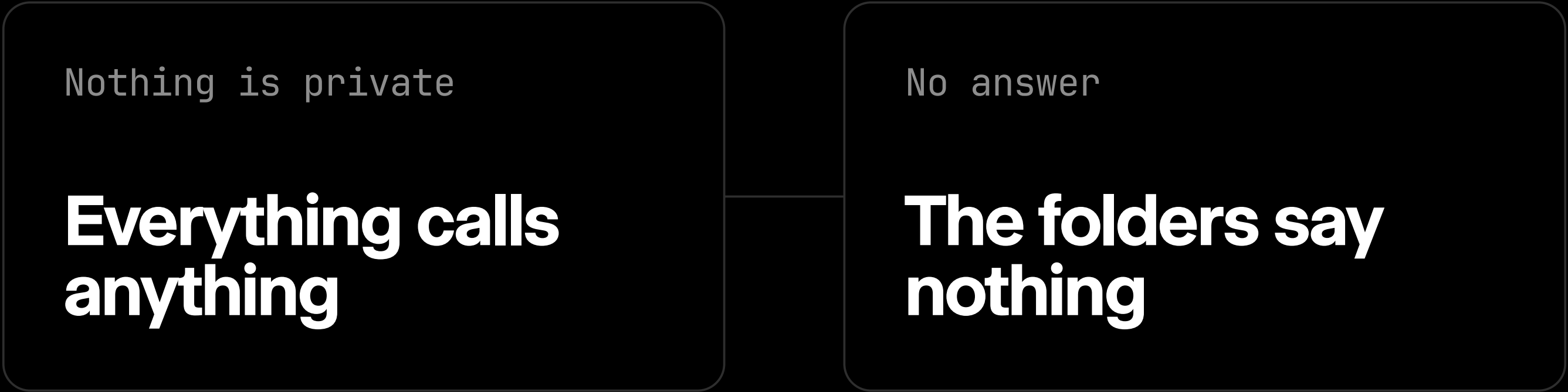
Shape

**Not
navig**

Shape

**Nothing to
navigate**

Flat: Cons



Nothing is private

**Everything calls
anything**

No ans

**The
noth**

No answer

**The folders say
nothing**

02

ORGANISED BY WHAT THE CODE DOES

By layer

1	02-layered/	
2	├── models/	
3	│ ├── occurrence.go	Occurrence, GBIF's shape
4	│ └── species.go	Species
5	├── gbif/	
6	│ └── gbif.go	API client
7	├── store/	
8	│ ├── store.go	Storage interface
9	│ └── mem.go	In-memory store
10	├── services/	
11	│ └── ingest.go	Ingest data & seed
12	├── handlers/	
13	│ ├── handler.go	GET /
14	│ └── sightings.html	HTML page
15	└── main.go	Wiring



1	02-layered/	
2	├── models/	
3	│ ├── occurrence.go	Occurrence, GBIF's shape
4	│ └── species.go	Species
5	├── gbif/	
6	│ └── gbif.go	API client
7	├── store/	
8	│ ├── store.go	Storage interface
9	│ └── mem.go	In-memory store
10	├── services/	
11	│ └── ingest.go	Ingest data & seed
12	├── handlers/	
13	│ ├── handler.go	GET /
14	│ └── sightings.html	HTML page
15	└── main.go	Wiring

```
1 02-layered/
2  └─ models/
3     └─ occurrence.go  Occurrence, GBIF's shape
4     └─ species.go    Species
5  └─ gbif/
6     └─ gbif.go        API client
7  └─ store/
8     └─ store.go       Storage interface
9     └─ mem.go         In-memory store
10 └─ services/
11    └─ ingest.go       Ingest data & seed
12 └─ handlers/
13    └─ handler.go      GET /
14    └─ sightings.html  HTML page
15 └─ main.go           Wiring
```

Three questions

Can you tell what it does?

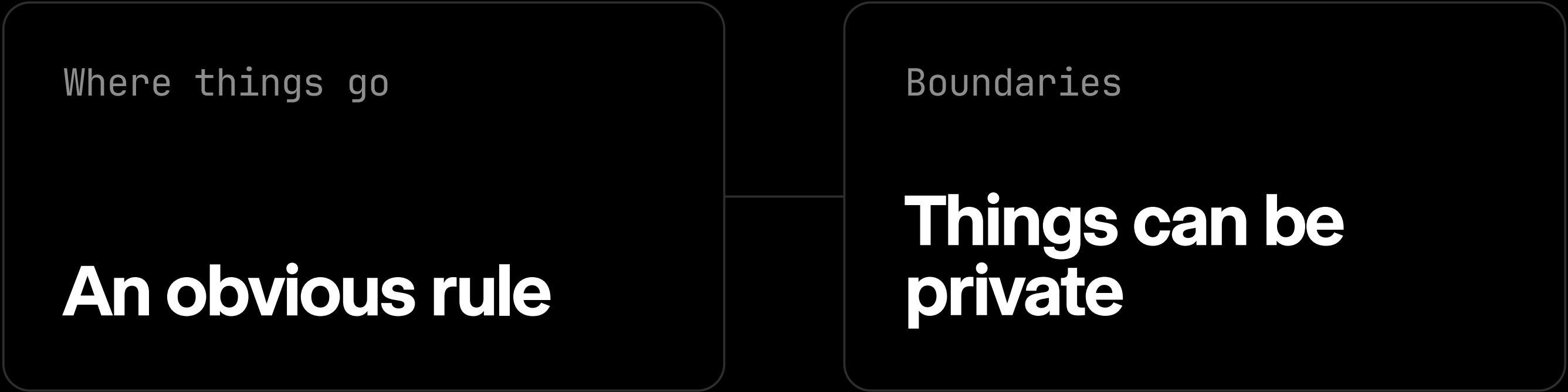
Where does a new feature go?

Where do the rules live?

```
1 // package models
2 func (o Occurrence) Save() error {
3     // import cycle: models → store → models
4     return store.SaveOccurrence(o)
5 }
```

```
1 // package models
2 func (o Occurrence) Save() error {
3     // import cycle: models → store → models
4     return store.SaveOccurrence(o)
5 }
```

By layer: Pros



Where things go

An obvious rule

Bounda

**Thir
priv**

Boundaries

**Things can be
private**

By layer: Cons

The cycle

**Models imports
nothing**

The spread

**One feature, four
folders**

The cycle

**Models imports
nothing**

The sp

**One
fold**

The spread

**One feature, four
folders**

03

ORGANISED BY RINGS

Onion

```
1 03-onion/
2  └─ domain/
3     └─ sighting.go      Sighting, our shape, and SightingStore
4     └─ species.go       Species, and SpeciesStore
5  └─ services/
6     └─ ingest.go        Ingest data & seed
7  └─ gbif/
8     └─ gbif.go          API client
9     └─ models.go        GBIF's shapes
10 └─ store/
11    └─ sightings.go      Implements domain.SightingStore
12    └─ species.go        Implements domain.SpeciesStore
13 └─ handlers/
14    └─ handler.go        GET /
15    └─ sightings.html    HTML page
16 └─ main.go              Wiring
```

1	03-onion/	
2	└─ domain/	
3	└─ sighting.go	Sighting, our shape, and SightingStore
4	└─ species.go	Species, and SpeciesStore
5	└─ services/	
6	└─ ingest.go	Ingest data & seed
7	└─ gbif/	
8	└─ gbif.go	API client
9	└─ models.go	GBIF's shapes
10	└─ store/	
11	└─ sightings.go	Implements domain.SightingStore
12	└─ species.go	Implements domain.SpeciesStore
13	└─ handlers/	
14	└─ handler.go	GET /
15	└─ sightings.html	HTML page
16	└─ main.go	Wiring

1	03-onion/	
2	└─ domain/	
3	└─ sighting.go	Sighting, our shape, and SightingStore
4	└─ species.go	Species, and SpeciesStore
5	└─ services/	
6	└─ ingest.go	Ingest data & seed
7	└─ gbif/	
8	└─ gbif.go	API client
9	└─ models.go	GBIF's shapes
10	└─ store/	
11	└─ sightings.go	Implements domain.SightingStore
12	└─ species.go	Implements domain.SpeciesStore
13	└─ handlers/	
14	└─ handler.go	GET /
15	└─ sightings.html	HTML page
16	└─ main.go	Wiring

1	03-onion/	
2	— domain/	
3	— sighting.go	Sighting, our shape, and SightingStore
4	— species.go	Species, and SpeciesStore
5	— services/	
6	— ingest.go	Ingest data & seed
7	— gbif/	
8	— gbif.go	API client
9	— models.go	GBIF's shapes
10	— store/	
11	— sightings.go	Implements domain.SightingStore
12	— species.go	Implements domain.SpeciesStore
13	— handlers/	
14	— handler.go	GET /
15	— sightings.html	HTML page
16	— main.go	Wiring

```
1 03-onion/
2  └─ domain/
3     └─ sighting.go      Sighting, our shape, and SightingStore
4     └─ species.go       Species, and SpeciesStore
5  └─ services/
6     └─ ingest.go        Ingest data & seed
7  └─ gbif/
8     └─ gbif.go           API client
9     └─ models.go        GBIF's shapes
10 └─ store/
11    └─ sightings.go      Implements domain.SightingStore
12    └─ species.go        Implements domain.SpeciesStore
13 └─ handlers/
14    └─ handler.go        GET /
15    └─ sightings.html    HTML page
16 └─ main.go              Wiring
```

Three questions

Can you tell what it does?

Where does a new feature go?

Where do the rules live?

Onion: Pros

Language

**Our own word for
it**

The arrow

**Storage answers
to the domain**

Language

Our own word for it

The an

Stor
the

or it

The arrow

**Storage answers to
the domain**

Onion: Cons

Empty middle

The domain has no rules

The drawer

Services takes anything

Empty middle

**The domain has no
rules**

The dr

**Serv
any**

s no

The drawer

**Services takes
anything**

s no

The drawer

**Services takes
anything**

04.

ORGANISED BY WHAT THE CODE IS ABOUT

Hexagonal

1	04-hexagonal/	
2	└─ domain/	
3	└─ sighting/	
4	└─ model.go	Sighting, our shape
5	└─ sighting.go	The Store port, the Service
6	└─ ingest.go	Ingest data & seed
7	└─ validate.go	No sightings from the future
8	└─ species/	Species, the rule, its own port
9	└─ clients/	
10	└─ gbif/	API client, GBIF's shapes
11	└─ stores/	
12	└─ sightingmem/	Implements sighting.Store
13	└─ speciesmem/	Implements species.Store
14	└─ handlers/	
15	└─ sightinghttp/	GET /
16	└─ web/	The page, and the shapes it draws
17	└─ main.go	Wiring and the server

1	04-hexagonal/	
2	└─ domain/	
3	└─ sighting/	
4	└─ model.go	Sighting, our shape
5	└─ sighting.go	The Store port, the Service
6	└─ ingest.go	Ingest data & seed
7	└─ validate.go	No sightings from the future
8	└─ species/	Species, the rule, its own port
9	└─ clients/	
10	└─ gbif/	API client, GBIF's shapes
11	└─ stores/	
12	└─ sightingmem/	Implements sighting.Store
13	└─ speciesmem/	Implements species.Store
14	└─ handlers/	
15	└─ sightinghttp/	GET /
16	└─ web/	The page, and the shapes it draws
17	└─ main.go	Wiring and the server

1	04-hexagonal/	
2	├── domain/	
3	│ ├── sighting/	
4	│ │ ├── model.go	Sighting, our shape
5	│ │ ├── sighting.go	The Store port, the Service
6	│ │ ├── ingest.go	Ingest data & seed
7	│ │ └── validate.go	No sightings from the future
8	│ └── species/	Species, the rule, its own port
9	├── clients/	
10	│ └── gbif/	API client, GBIF's shapes
11	├── stores/	
12	│ ├── sightingmem/	Implements sighting.Store
13	│ └── speciesmem/	Implements species.Store
14	├── handlers/	
15	│ └── sightinghttp/	GET /
16	├── web/	The page, and the shapes it draws
17	└── main.go	Wiring and the server

1	04-hexagonal/	
2	└─ domain/	
3	└─ sighting/	
4	└─ model.go	Sighting, our shape
5	└─ sighting.go	The Store port, the Service
6	└─ ingest.go	Ingest data & seed
7	└─ validate.go	No sightings from the future
8	└─ species/	Species, the rule, its own port
9	└─ clients/	
10	└─ gbif/	API client, GBIF's shapes
11	└─ stores/	
12	└─ sightingmem/	Implements sighting.Store
13	└─ speciesmem/	Implements species.Store
14	└─ handlers/	
15	└─ sightinghttp/	GET /
16	└─ web/	The page, and the shapes it draws
17	└─ main.go	Wiring and the server

1	04-hexagonal/	
2	└─ domain/	
3	└─ sighting/	
4	└─ model.go	Sighting, our shape
5	└─ sighting.go	The Store port, the Service
6	└─ ingest.go	Ingest data & seed
7	└─ validate.go	No sightings from the future
8	└─ species/	Species, the rule, its own port
9	└─ clients/	
10	└─ gbif/	API client, GBIF's shapes
11	└─ stores/	
12	└─ sightingmem/	Implements sighting.Store
13	└─ speciesmem/	Implements species.Store
14	└─ handlers/	
15	└─ sightinghttp/	GET /
16	└─ web/	The page, and the shapes it draws
17	└─ main.go	Wiring and the server

Three questions

Can you tell what it does?

Where does a new feature go?

Where do the rules live?

Hexagonal: Pros

An answer

**The folders name
the nouns**

Invariants

**The rules have a
home**

An answer

**The folders name
the nouns**

Invar:

**The
hom**

Invariants

**The rules have a
home**

Hexagonal: Cons

The count

**Seven packages,
one list**

02's words

**Stores and
handlers naming**

The count

**Seven packages,
one list**

02's v

**Stor
han**

02's words

Stores and handlers naming

05

ORGANISED BY THE FEATURE

Feature

```
1 05-feature/
2  └── domain/
3      ├── sighting/
4          ├── model.go
5          ├── sighting.go
6          ├── ingest.go
7          ├── validate.go
8          └── stores/sightingmem/
9      └── species/
10         ├── species.go
11         └── stores/speciesmem/
12 └── clients/gbif/
13 └── web/
14     ├── web.go
15     ├── handlers/home.go
16     └── views/
17 └── main.go
```

Sighting, carrying its species
The ports, the Service
Ingest data & seed
A species, and not the future
Implements sighting.Store

The Store port, the Service
Implements species.Store

The API client

New() – the ServeMux, the only URLs
GET /, a function returning a handler
The templates
Wiring





```
1 // package sighting
2 type SpeciesReadWriter interface {
3     Find(key int) (species.Species, error)
4     Save(sp species.Species) error
5 }
```

```
1 // package sighting
2 type SpeciesReadWriter interface {
3     Find(key int) (species.Species, error)
4     Save(sp species.Species) error
5 }
```

```
1 // package sighting
2 type SpeciesReadWriter interface {
3     Find(key int) (species.Species, error)
4     Save(sp species.Species) error
5 }
```

Three questions

Can you tell what it does?

Where does a new feature go?

Where do the rules live?

Feature: Pros

One place

**A feature is a
folder**

Scale

**The root stops
growing**

One place

A feature is a folder

Scale

**The
grow**

lder

Scale

**The root stops
growing**

Feature: Cons



Typing

The paths are long

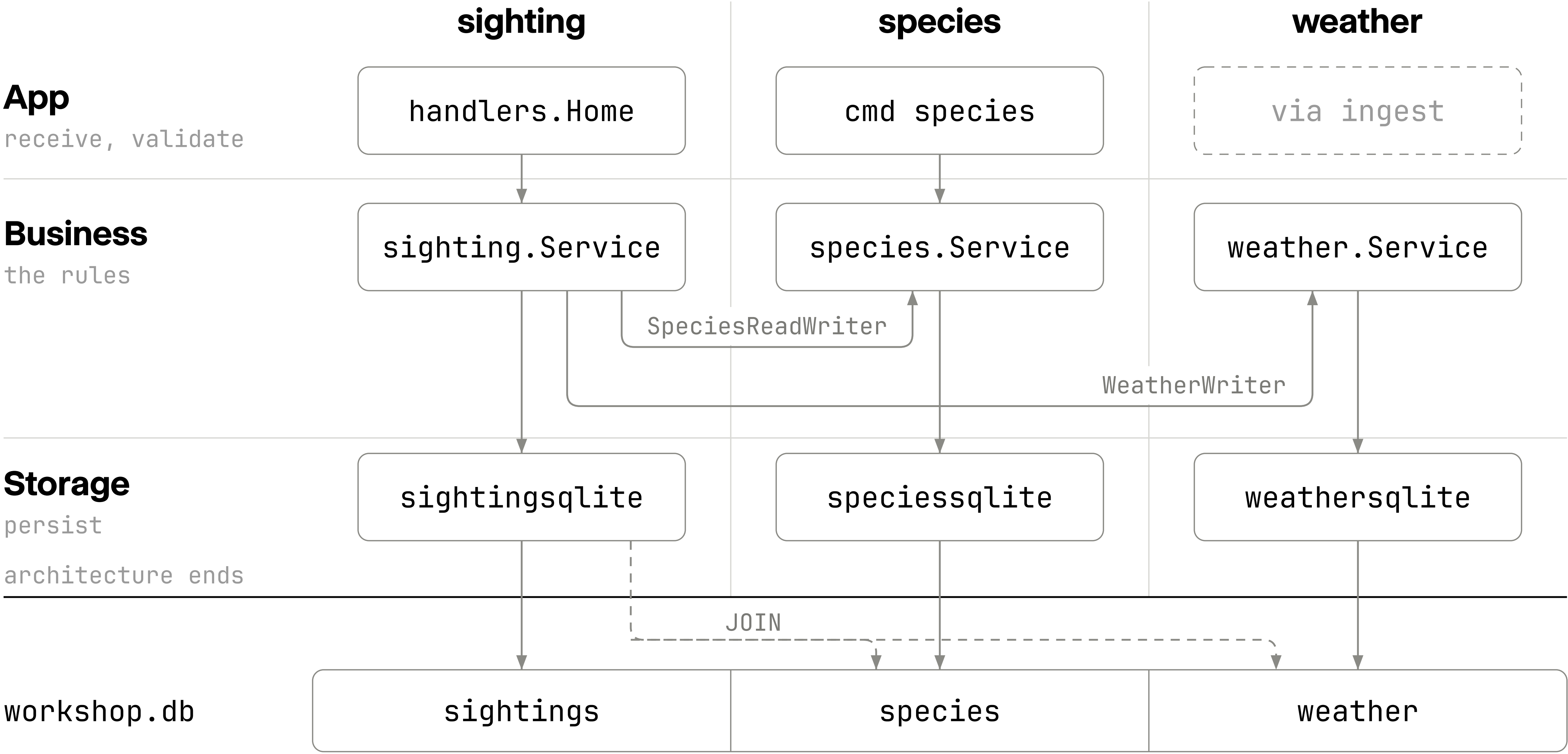
The ca

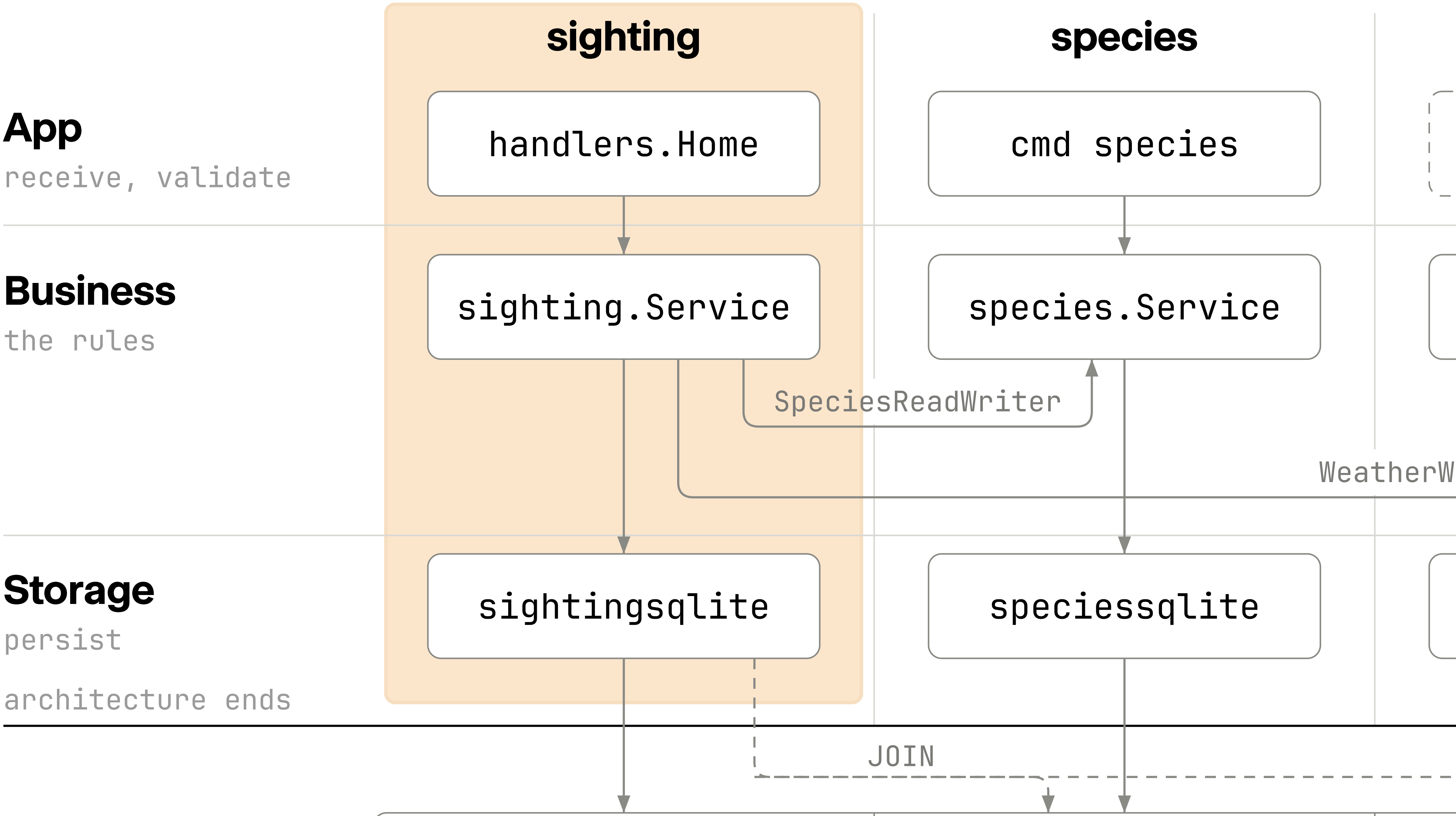
**Two
rem**

The caveat

**Two rules to
remember**

ng

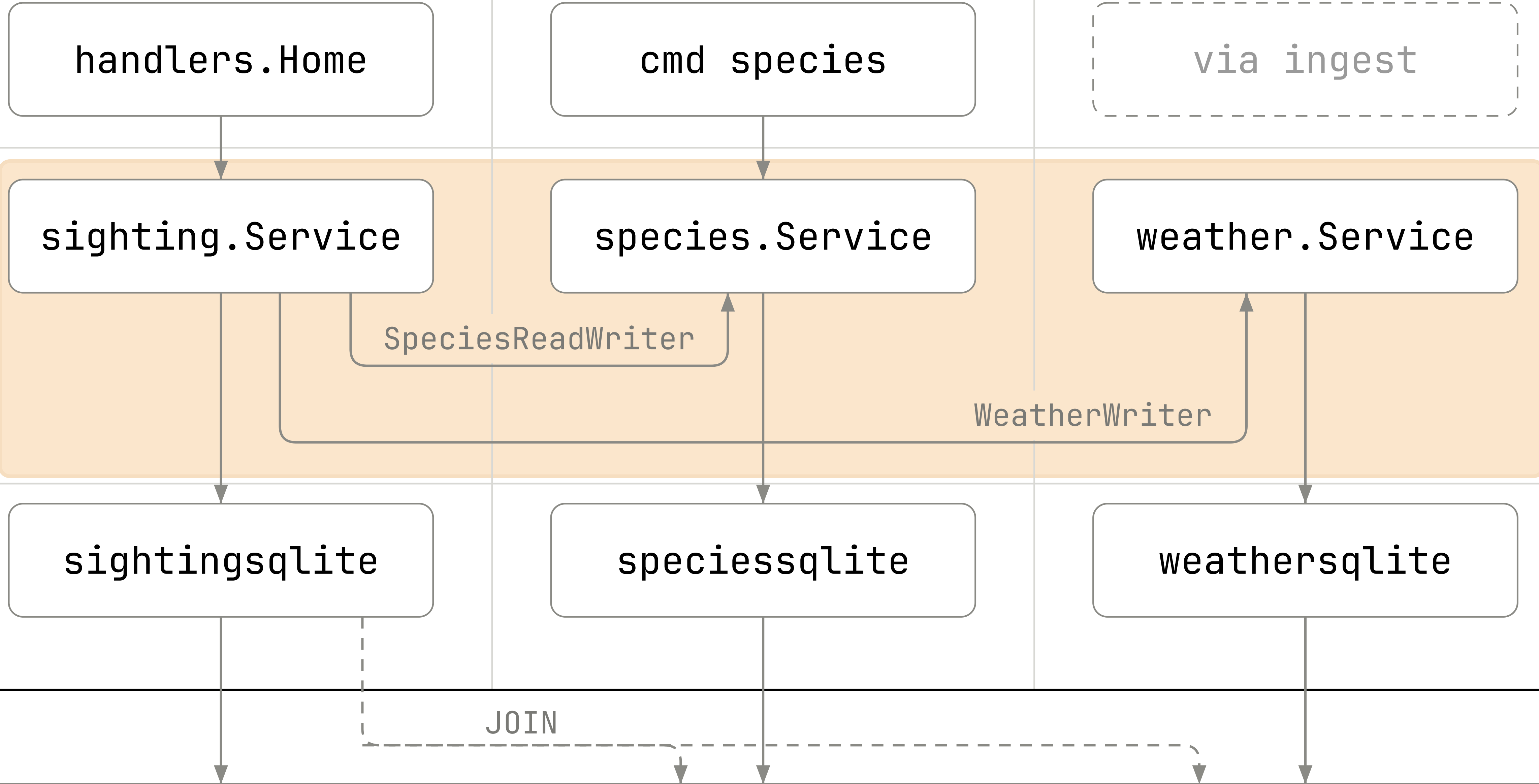




sighting

species

weather



Business

Abstract the rules

sighting.Service

species.Service

weather.Service

SpeciesReadWriter

WeatherWriter

Storage

persist

sightingsqlite

speciessqlite

weathersqlite

architecture ends

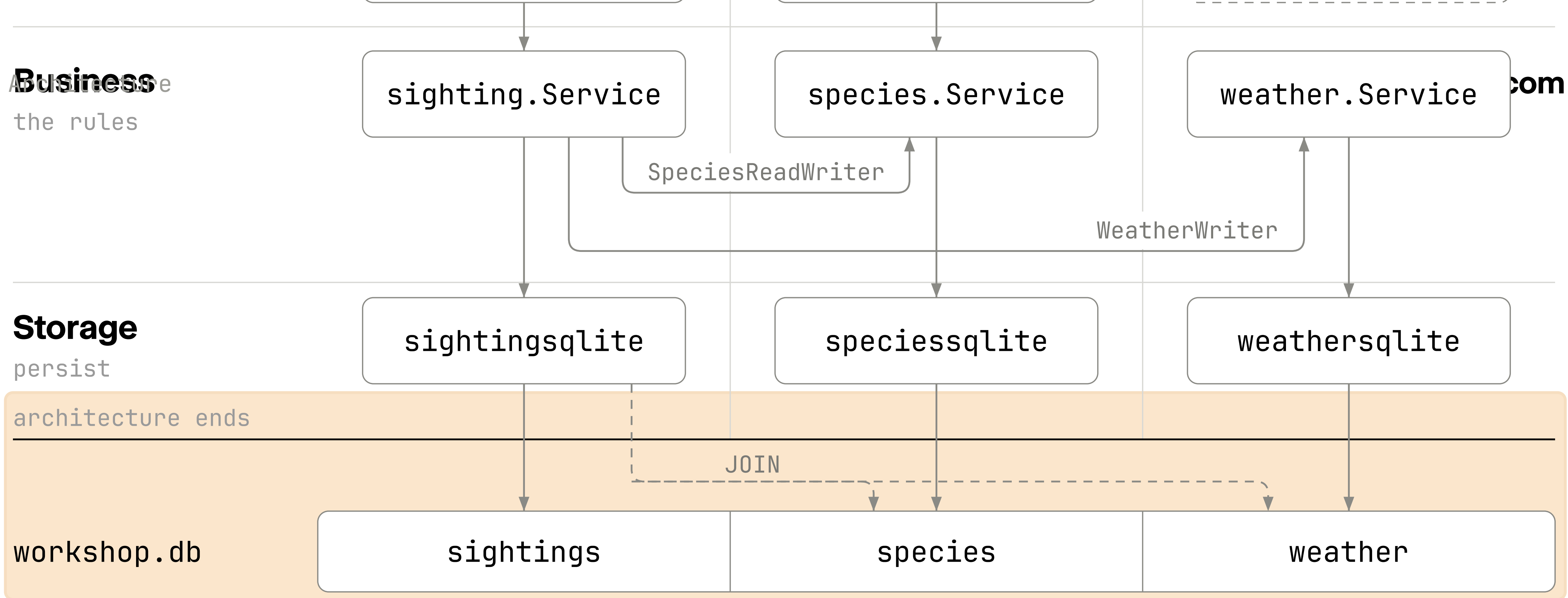
JOIN

workshop.db

sightings

species

weather



Architecture

SpeciesReadWriter

WeatherWriter

ainsleyclark.com

sightingsqlite

speciessqlite

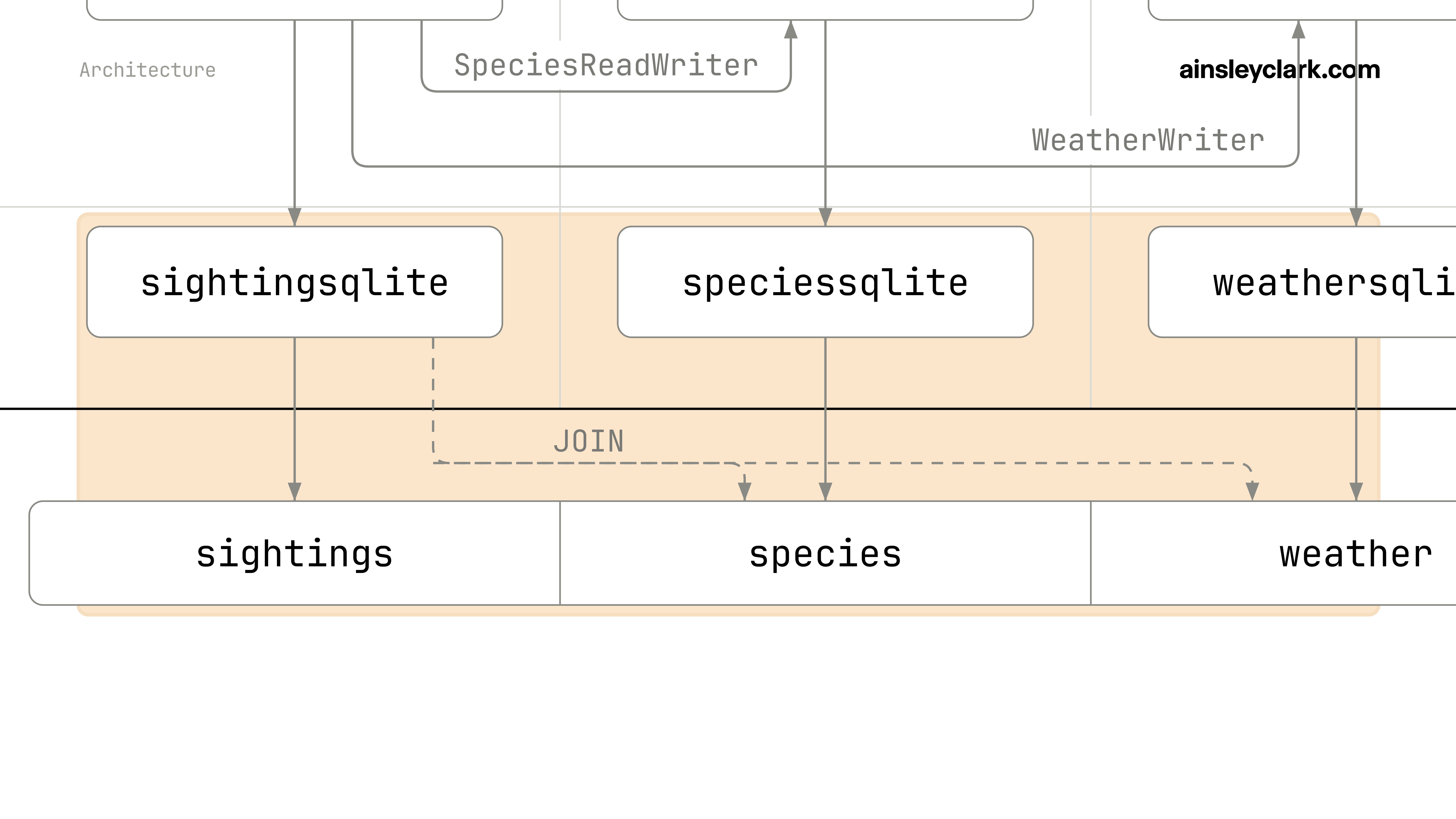
weathersqli

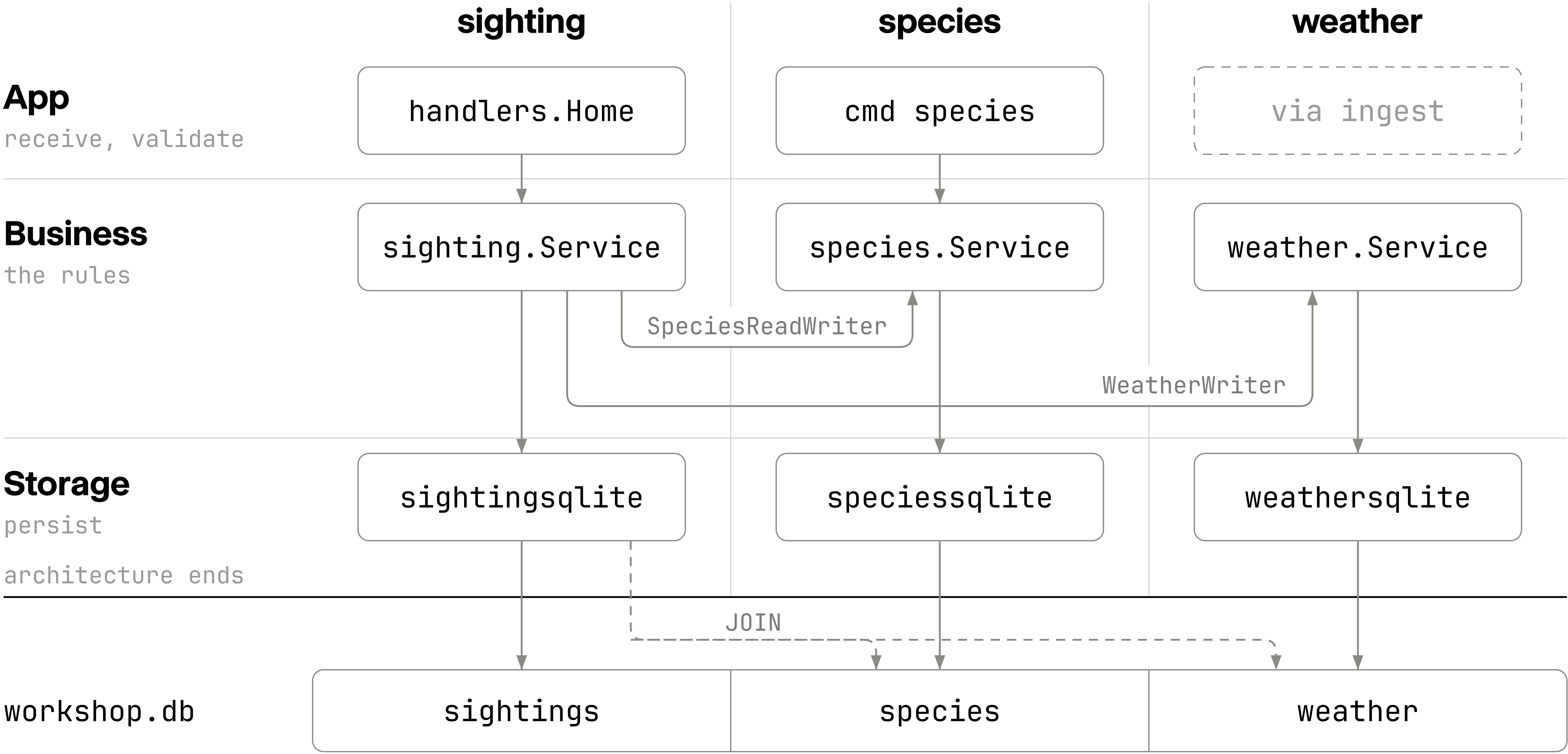
JOIN

sightings

species

weather





DDD thoughts

The rule of five

**The structure should tell the
truth**

**The structure should tell the
truth**

I never got it right first time

I never got it right first time

I never got it right first time

This is what makes you valuable

App tour

Your turn

Slack



Three months of GoLand, free GopherConAfrica2026

Code: GopherConAfrica2026

Create an account at account.jetbrains.com

Redeem at jetbrains.com/store/redeem

Open GoLand and log in to your JetBrains Account

Refresher

<https://sqlc.dev>

Querying with sqlc

sqlc

What?

Why?

When?

Migrations

```
1  -- migrations/0001_init.sql
2  -- +goose Up
3  -- +goose StatementBegin
4  CREATE TABLE species (
5      id UUID PRIMARY KEY,
6      "gbif_key" INTEGER NOT NULL UNIQUE,
7      "kingdom" TEXT NOT NULL
8  );
9  -- +goose StatementEnd
10
11 -- +goose Down
12 -- +goose StatementBegin
13 DROP TABLE IF EXISTS species;
14 -- +goose StatementEnd
```

```
1  -- migrations/0001_init.sql
2  -- +goose Up
3  -- +goose StatementBegin
4  CREATE TABLE species (
5      id UUID PRIMARY KEY,
```

```
2      -- +goose Up
3      -- +goose StatementBegin
4      CREATE TABLE species (
5          id UUID PRIMARY KEY,
6          "gbif_key" INTEGER NOT NULL UNIQUE,
7          "kingdom" TEXT NOT NULL
8      );
9      -- +goose StatementEnd
10
```

10

11 -- +goose Down

12 -- +goose StatementBegin

13 **DROP TABLE IF EXISTS** species;

14 -- +goose StatementEnd

go tool sqlc generate

```
1 // internal/infra/db/sqlc/models.go
2 type Species struct {
3     ID          uuid.UUID `db:"id"`
4     GbifKey      int64     `db:"gbif_key"`
5     Kingdom      string     `db:"kingdom"`
6 }
```

Queries

```
1  -- queries/species.sql
2  -- name: SpeciesFind :one
3  SELECT * FROM species WHERE id = ?;
4
5  -- name: SpeciesList :many
6  SELECT * FROM species;
7
8  -- name: SpeciesCreate :one
9  INSERT INTO species (id, "gbif_key", "kingdom")
10 VALUES (?, ?, ?) RETURNING *;
```

```
1  -- queries/species.sql
2  -- name: SpeciesFind :one
3  SELECT * FROM species WHERE id = ?;
4
5  -- name: SpeciesList :many
6  SELECT * FROM species;
```

```
1  -- queries/species.sql
2  -- name: SpeciesFind :one
3  SELECT * FROM species WHERE id
4
5  -- name: SpeciesList :many
```

```
1  -- queries/species.sql
2  -- name: SpeciesFind :one
3  SELECT * FROM species WHERE id = ?;
4
5  -- name: SpeciesList :many
6  SELECT * FROM species;
7
```

```
3      SELECT * FROM species WHERE id
4
5      -- name: SpeciesList :many
6      SELECT * FROM species;
7
8      -- name: SpeciesCreate :one
```

```
4
5  -- name: SpeciesList :many
6  SELECT * FROM species;
7
8  -- name: SpeciesCreate :one
9  INSERT INTO species (id, "gbif_key", "kingdom")
10 VALUES (?, ?, ?) RETURNING *;
```

```
1 // internal/infra/db/sqlc/species.sql.go
2 func (q Queries) SpeciesFind(ctx context.Context, id uuid.UUID) (Species, error) {
3     row := q.db.QueryRowContext(ctx, speciesFind, id)
4     var i Species
5     err := row.Scan(
6         &i.ID,
7         &i.GbifKey,
8         &i.Kingdom,
9     )
10    return i, err
11 }
```

```
1 // internal/infra/db/sqlc/species.sql.go
2 func (q Queries) SpeciesFind(ctx context.Context) (Species, error) {
3     row := q.db.QueryRowContext(ctx, speciesFindQuery)
4     var i Species
5     err := row.Scan(&i)
```

```
1 // internal/infra/db/sqlc/species.sql.go
2 func (q Queries) SpeciesFind(ctx context.Context, id uuid.UUID) (Species, error) {
3     row := q.db.QueryRowContext(ctx, speciesFind, id)
4     var i Species
5     err := row.Scan(
6         &i.ID,
7         &i.GbifKey,
8         &i.Kingdom,
9     )
10    return i, err
11 }
```

```
1 // internal/infra/db/sqlc/species.sql.go
2 func (q Queries) SpeciesFind(ctx context.Context, id uuid.UUID) (*Species, error) {
3     row := q.db.QueryRowContext(ctx, speciesFind, id)
4     var i Species
5     err := row.Scan(
6         &i.ID,
7         &i.GbifKey,
8         &i.Kingdom,
9     )
```

```
4      var i Species
5      err := row.Scan(
6          &i.ID,
7          &i.GbifKey,
8          &i.Kingdom,
9      )
10     return i, err
11 }
```

```
1 // internal/infra/db/sqlc/species.sql.go
2 func (q Queries) SpeciesFind(ctx context.Context, id uuid.UUID) (Species, error) {
3     row := q.db.QueryRowContext(ctx, speciesFind, id)
4     var i Species
5     err := row.Scan(
6         &i.ID,
7         &i.GbifKey,
8         &i.Kingdom,
9     )
10    return i, err
11 }
```

```
1 // internal/infra/db/sqlc/querier.go
2 type Querier interface {
3     SpeciesFind(ctx context.Context, id uuid.UUID) (Species, error)
4     SpeciesList(ctx context.Context) ([]Species, error)
5     SpeciesCreate(ctx context.Context, arg SpeciesCreateParams) (Species, error)
6 }
```

sqlc.yaml

DB Testing

Demo

Your turn

<https://templ.guide>

Intro to templ

 *TEMPL*

What?

What?

Why?

Why?

When?

Common theme?

Fundamentals

```
1  // web/views/layout/layout.temp1
2  package layout
3
4  type BaseProps struct {
5      Title string
6  }
7
8  templ Base(props BaseProps) {
9      <!DOCTYPE html>
10     <html lang="en-GB">
11         <head>
12             <title>{ props.Title }</title>
13         </head>
14         <body>
15             <main class="container">
16                 { children... }
17             </main>
18         </body>
19     </html>
20 }
```

```
1  // web/views/layout/layout.temp1
2  package layout
3
4  type BaseProps struct {
5      Title string
6  }
7
8  templ Base(props BaseProps) {
9      <!DOCTYPE html>
10     <html lang="en-GB">
11         <head>
12             <title>{ props.Title }</title>
13         </head>
14         <body>
15             <main class="container">
16                 { children... }
17             </main>
18         </body>
19     </html>
20 }
```

```
1 // web/views/layout/layout.tpl
2 package layout
3
4 type BaseProps struct {
5     Title string
```

3

4

```
type BaseProps struct {
```

5

```
    Title string
```

6

```
}
```

7

8

```
templ Base(props BaseProps) +
```

```
5      Title string
6  }
7
8  templ Base(props BaseProps) {
9      <!DOCTYPE html>
10     <html lang="en-GB">
11         <head>
```

```
8      templ Base(props BaseProps) {
9          <!DOCTYPE html>
10         <html lang="en-GB">
11             <head>
12                 <title>{ props.Title }</title>
13             </head>
14             <body>
15                 <main class="container">
16                     { children... }
17                 </main>
```

```
13      </head>
14      <body>
15          <main class="container">
16              { children... }
17          </main>
18      </body>
19  </html>
```

go tool templ generate ./...

```
1 // web/views/pages/home.tpl
2 package pages
3
4 import (
5     "workshop/web/views/layout"
6 )
7
8 tmpl Home() {
9     @layout.Base(layout.BaseProps{
10         Title: "Sightings",
11     }) {
12         <h1>Sightings</h1>
13     }
14 }
```

```
1 // web/views/pages/home.temp  
2 package pages  
3  
4 import (
```

```
2 package pages
3
4 import (
5     "workshop/web/views/layout"
6 )
7
8 templ Home() {
```

```
7
8   templ Home() {
9       @layout.Base(layout.BaseProps{
10           Title: "Sightings",
11       }) {
12           <h1>Sightings</h1>
13       }
14   }
```

```
9      @Layout.Base(Layout.BasePro
10          Title: "Sightings",
11      }) {
12          <h1>Sightings</h1>
13      }
14 }
```

```
1 // web/views/pages/home.tpl
2 package pages
3
4 import (
5     "workshop/web/views/layout"
6 )
7
8 tmpl Home() {
9     @layout.Base(layout.BaseProps{
10         Title: "Sightings",
11     }) {
12         <h1>Sightings</h1>
13     }
14 }
```

```
1  // web/views/components
2  templ Gallery(items []sighting.Media) {
3      <ul class="gallery">
4          for _, m := range items {
5              <li><img src={ m.URL } alt={ altText(m) }/></li>
6          }
7      </ul>
8  }
9
10 templ SightingCard(s sighting.Sighting) {
11     <article class="card">
12         <h2>{ displayName(s.Species) }</h2>
13         @Gallery(s.Media)
14     </article>
15 }
```

```
1 // web/views/components
2 templ Gallery(items []sighting.Media) {
3     <ul class="gallery">
4         for _, m := range items {
5             <li><img src={ m.URL } alt={ altText(m) }/></li>
6         }
7     </ul>
8 }
9
10 templ SightingCard(s sighting.Sighting) {
11     <article class="card">
12         <h2>{ displayName(s.Species) }</h2>
```

```
8      }  
9  
10     templ SightingCard(s sighting.Sighting) {  
11         <article class="card">  
12             <h2>{ displayName(s.Species) }</h2>  
13             @Gallery(s.Media)  
14         </article>  
15     }
```

```
11      <article class="card">
12          <h2>{ displayName(s.Sp
13              @Gallery(s.Media)
14      </article>
15  }
```

```
1  // web/views/components
2  templ Gallery(items []sighting.Media) {
3      <ul class="gallery">
4          for _, m := range items {
5              <li><img src={ m.URL } alt={ altText(m) }/></li>
6          }
7      </ul>
8  }
9
10 templ SightingCard(s sighting.Sighting) {
11     <article class="card">
12         <h2>{ displayName(s.Species) }</h2>
13         @Gallery(s.Media)
14     </article>
15 }
```

```
1 // web/handlers/home.go
2 type SightingsLister interface {
3     List(ctx context.Context, f sighting.ListFilter) ([]sighting.Sighting, error)
4 }
5
6 func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7     return func(w http.ResponseWriter, r *http.Request) {
8         ctx := r.Context()
9
10        items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11        if err != nil {
12            renderError(ctx, w, logger, 500, "Unable to load sightings.")
13            return
14        }
15        w.Header().Set("Content-Type", "text/html; charset=utf-8")
16        if err = pages.Home(items).Render(ctx, w); err != nil {
17            renderError(ctx, w, logger, 500, "Unable to render page.")
18        }
19    }
20 }
```

```
1 // web/handlers/home.go
2 type SightingsLister interface {
3     List(ctx context.Context, f sighting.ListFilter) ([]sighting.Sighting, error)
4 }
5
6 func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7     return func(w http.ResponseWriter, r *http.Request) {
8         ctx := r.Context()
9
10        items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11        if err != nil {
12            renderError(ctx, w, logger, 500, "Unable to load sightings.")
13            return
14        }
15        w.Header().Set("Content-Type", "text/html; charset=utf-8")
16        if err = pages.Home(items).Render(ctx, w); err != nil {
17            renderError(ctx, w, logger, 500, "Unable to render page.")
18        }
19    }
20 }
```

```
1 // web/handlers/home.go
2 type SightingsLister interface {
3     List(ctx context.Context)
4 }
```

```
1 // web/handlers/home.go
2 type SightingsLister interface {
3     List(ctx context.Context, f sighting.ListFilter) ([]sighting.Sighting, error)
4 }
5
6 func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7     return func(w http.ResponseWriter, r *http.Request) {
8         ctx := r.Context()
9
10        items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11        if err != nil {
12            renderError(ctx, w, logger, 500, "Unable to load sightings.")
13        }
14    }
15 }
```

```
1 // web/handlers/home.go
2 type SightingsLister interface {
3     List(ctx context.Context, f sighting.ListFilter) ([]sighting.Sighting, error)
4 }
5
6 func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7     return func(w http.ResponseWriter, r *http.Request) {
8         ctx := r.Context()
9
10        items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11        if err != nil {
12            renderError(ctx, w, logger, 500, "Unable to load sightings.")
13            return
14        }
15        w.Header().Set("Content-Type", "text/html; charset=utf-8")
```

```
2  type SightingsLister interface {
3      List(ctx context.Context, f sighting.ListFilter) ([]sighting.Sighting, error)
4  }
5
6  func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7      return func(w http.ResponseWriter, r *http.Request) {
8          ctx := r.Context()
9
10         items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11         if err != nil {
12             renderError(ctx, w, logger, 500, "Unable to load sightings.")
13             return
14         }
15         w.Header().Set("Content-Type", "text/html; charset=utf-8")
16         if err = pages.Home(items).Render(ctx, w); err != nil {
17             renderError(ctx, w, logger, 500, "Unable to render page.")
18         }
19     }
20 }
```

```
4     }
5
6     func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7         return func(w http.ResponseWriter, r *http.Request) {
8             ctx := r.Context()
9
10            items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11            if err != nil {
12                renderError(ctx, w, logger, 500, "Unable to load sightings.")
13                return
14            }
15            w.Header().Set("Content-Type", "text/html; charset=utf-8")
16            if err = pages.Home(items).Render(ctx, w); err != nil {
17                renderError(ctx, w, logger, 500, "Unable to render page.")
18            }
19        }
20    }
```

```
9
10     items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11     if err != nil {
12         renderError(ctx, w, logger, 500, "Unable to load sightings.")
13         return
14     }
15     w.Header().Set("Content-Type", "text/html; charset=utf-8")
16     if err = pages.Home(items).Render(ctx, w); err != nil {
17         renderError(ctx, w, logger, 500, "Unable to render page.")
18     }
19 }
20 }
```

```
1 // web/handlers/home.go
2 type SightingsLister interface {
3     List(ctx context.Context, f sighting.ListFilter) ([]sighting.Sighting, error)
4 }
5
6 func Home(logger *slog.Logger, sightings SightingsLister) http.HandlerFunc {
7     return func(w http.ResponseWriter, r *http.Request) {
8         ctx := r.Context()
9
10        items, err := sightings.List(ctx, sighting.ListFilter{Limit: 50})
11        if err != nil {
12            renderError(ctx, w, logger, 500, "Unable to load sightings.")
13            return
14        }
15        w.Header().Set("Content-Type", "text/html; charset=utf-8")
16        if err = pages.Home(items).Render(ctx, w); err != nil {
17            renderError(ctx, w, logger, 500, "Unable to render page.")
18        }
19    }
20 }
```

Tooling

Demo

Your turn

Profiling

What?

Why?

When?

How?

How?

Demo

Theming

Make it your own

Air

Demo

Ideas

Add a page per sighting (/sightings/{id}) or animal or country

Show the weather on each sighting

Add a statistics page

Put the sightings on a map

Add a photo gallery

Ask the skill to add some 3D effects

Come up with some more data points & migrations

Wrapping up

git push

make run-live

Competitions

Best website: pushed to your fork before you leave

Best social post: tag @Ainsley Clark, @JetBrains, @GopherCon Africa

Showcase?

Continue to build...

Photo

Thanks

