Flutter Architecture Components

By Alessio Salvadorini

@ASalvadorini
www.fluxit.dev

Intro

How it started ...

Google I/O 2018 -> BLoC
(source)

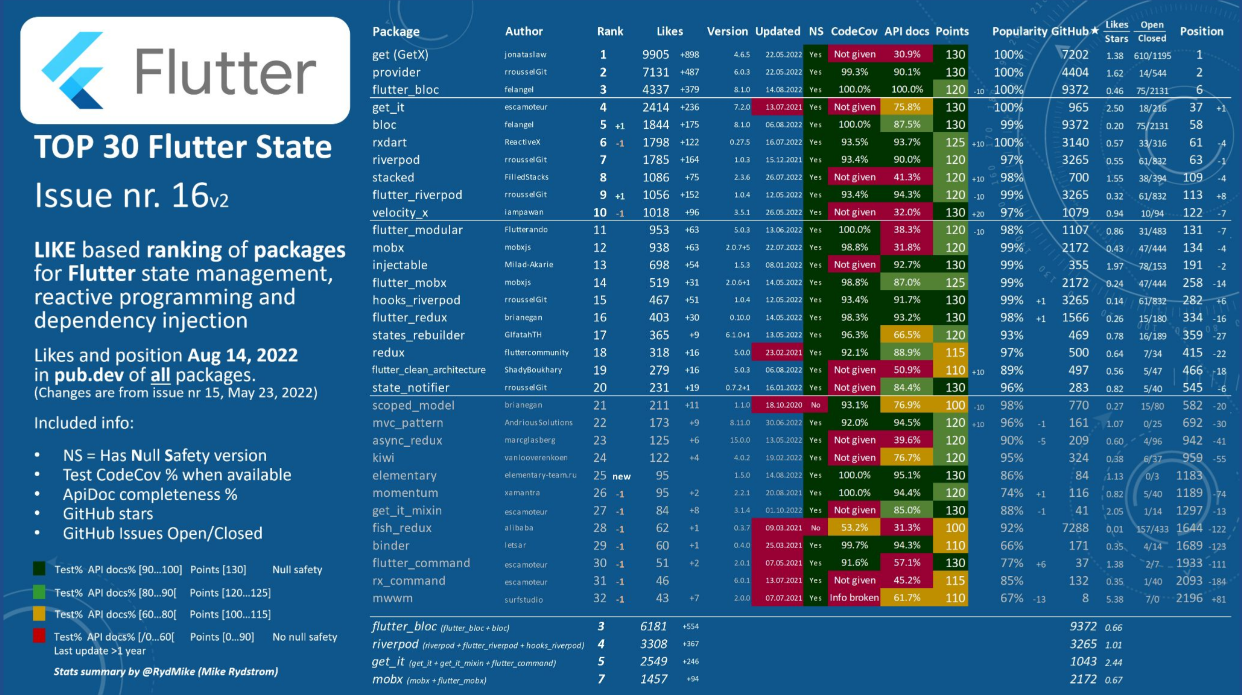
Google I/O 2019 -> Provider
(source)

How it is going

- Provider/Riverpod
- setState()
- InheritedWidget
- Redux
- BLoC
- GetIt
- GetX
- Triple Pattern
- … and more

(source)

How it is going



Flutter

TOP 30 Flutter State

Issue nr. 16v2

LIKE based **ranking** of **packages** for **Flutter** state management, reactive programming and dependency injection

Likes and position **Aug 14, 2022** in **pub.dev** of **all** packages.
(Changes are from issue nr 15, May 23, 2022)

Included info:

- NS = Has **N**ull **S**afety version
- Test CodeCov % when available
- ApiDoc completeness %
- GitHub stars
- GitHub Issues Open/Closed

🟩	Test% API docs% [90...100]	Points [130]	Null safety
🟩	Test% API docs% [80...90[	Points [120...125]	
🟨	Test% API docs% [60...80[	Points [100...115]	
🟥	Test% API docs% [/0...60[Last update >1 year	Points [0...90]	No null safety

Stats summary by @RydMike (Mike Rydstrom)

Package	Author	Rank		Likes		Version	Updated	NS	CodeCov	API docs	Points		Popularity		GitHub★	Likes/Stars	Open/Closed	Position	
get (GetX)	jonataslaw	1		9905	+898	4.6.5	22.05.2022	Yes	Not given	30.9%	130		100%		7202	1.38	610/1195	1	
provider	rrousselGit	2		7131	+487	6.0.3	22.05.2022	Yes	99.3%	90.1%	130		100%		4404	1.62	14/544	2	
flutter_bloc	felangel	3		4337	+379	8.1.0	14.08.2022	Yes	100.0%	100.0%	120	-10	100%		9372	0.46	75/2131	6	
get_it	escamoteur	4		2414	+236	7.2.0	13.07.2021	Yes	Not given	75.8%	130		100%		965	2.50	18/216	37	+1
bloc	felangel	5	+1	1844	+175	8.1.0	06.08.2022	Yes	100.0%	87.5%	130		99%		9372	0.20	75/2131	58	
rxdart	ReactiveX	6	-1	1798	+122	0.27.5	16.07.2022	Yes	93.5%	93.7%	125	+10	100%		3140	0.57	33/316	61	-4
riverpod	rrousselGit	7		1785	+164	1.0.3	15.12.2021	Yes	93.4%	90.0%	120		97%		3265	0.55	61/832	63	-1
stacked	FilledStacks	8		1086	+75	2.3.6	26.07.2022	Yes	Not given	41.3%	120	+10	98%		700	1.55	38/394	109	-4
flutter_riverpod	rrousselGit	9	+1	1056	+152	1.0.4	12.05.2022	Yes	93.4%	94.3%	120	-10	99%		3265	0.32	61/832	113	+8
velocity_x	iampawan	10	-1	1018	+96	3.5.1	26.05.2022	Yes	Not given	32.0%	130	+20	97%		1079	0.94	10/94	122	-7
flutter_modular	Flutterando	11		953	+63	5.0.3	13.06.2022	Yes	100.0%	38.3%	120	-10	98%		1107	0.86	31/483	131	-7
mobx	mobxjs	12		938	+63	2.0.7+5	22.07.2022	Yes	98.8%	31.8%	120		99%		2172	0.43	47/444	134	-4
injectable	Milad-Akarie	13		698	+54	1.5.3	08.01.2022	Yes	Not given	92.7%	130		99%		355	1.97	78/153	191	-2
flutter_mobx	mobxjs	14		519	+31	2.0.6+1	14.05.2022	Yes	98.8%	87.0%	125		99%		2172	0.24	47/444	258	-14
hooks_riverpod	rrousselGit	15		467	+51	1.0.4	12.05.2022	Yes	93.4%	91.7%	130		99%	+1	3265	0.14	61/832	282	+6
flutter_redux	brianegan	16		403	+30	0.10.0	14.05.2022	Yes	98.3%	93.2%	130		98%	+1	1566	0.26	15/180	334	-16
states_rebuilder	GIfatahTH	17		365	+9	6.1.0+1	13.05.2022	Yes	96.3%	66.5%	120		93%		469	0.78	16/189	359	-27
redux	fluttercommunity	18		318	+16	5.0.0	23.02.2021	Yes	92.1%	88.9%	115		97%		500	0.64	7/34	415	-22
flutter_clean_architecture	ShadyBoukhary	19		279	+16	5.0.3	06.08.2022	Yes	Not given	50.9%	110	+10	89%		497	0.56	5/47	466	-18
state_notifier	rrousselGit	20		231	+19	0.7.2+1	16.01.2022	Yes	Not given	84.4%	130		96%		283	0.82	5/40	545	-6
scoped_model	brianegan	21		211	+11	1.1.0	18.10.2020	No	93.1%	76.9%	100	-10	98%		770	0.27	15/80	582	-20
mvc_pattern	AndriousSolutions	22		173	+9	8.11.0	30.06.2022	Yes	92.0%	94.5%	120	+10	96%	-1	161	1.07	0/25	692	-30
async_redux	marcglasberg	23		125	+6	15.0.0	13.05.2022	Yes	Not given	39.6%	120		90%	-5	209	0.60	4/96	942	-41
kiwi	vanlooverenkoen	24		122	+4	4.0.2	19.02.2022	Yes	Not given	76.7%	120		95%		324	0.38	6/37	959	-55
elementary	elementary-team.ru	25	new	95		1.5.0	14.08.2022	Yes	100.0%	95.1%	130		86%		84	1.13	0/3	1183	
momentum	xamantra	26	-1	95	+2	2.2.1	20.08.2021	Yes	100.0%	94.4%	120		74%	+1	116	0.82	5/40	1189	-74
get_it_mixin	escamoteur	27	-1	84	+8	3.1.4	01.10.2022	Yes	Not given	85.0%	130		88%	-1	41	2.05	1/14	1297	-13
fish_redux	alibaba	28	-1	62	+1	0.3.7	09.03.2021	No	53.2%	31.3%	100		92%		7288	0.01	157/433	1644	-122
binder	letsar	29	-1	60	+1	0.4.0	25.03.2021	Yes	99.7%	94.3%	110		66%		171	0.35	4/14	1689	-123
flutter_command	escamoteur	30	-1	51	+2	2.0.1	07.05.2021	Yes	91.6%	57.1%	130		77%	+6	37	1.38	2/7	1933	-111
rx_command	escamoteur	31	-1	46		6.0.1	13.07.2021	Yes	Not given	45.2%	115		85%		132	0.35	1/40	2093	-184
mwwm	surfstudio	32	-1	43	+7	2.0.0	07.07.2021	Yes	Info broken	61.7%	110		67%	-13	8	5.38	7/0	2196	+81
flutter_bloc (flutter_bloc + bloc)		*3*		*6181*	+554										*9372*	*0.66*			
riverpod (riverpod + flutter_riverpod + hooks_riverpod)		*4*		*3308*	+367										*3265*	*1.01*			
get_it (get_it + get_it_mixin + flutter_command)		*5*		*2549*	+246										*1043*	*2.44*			
mobx (mobx + flutter_mobx)		*7*		*1457*	+94										*2172*	*0.67*			

Mike's comparison (source)

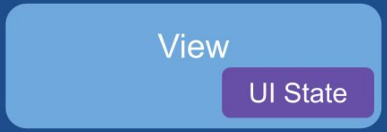
State Management

When to use state management

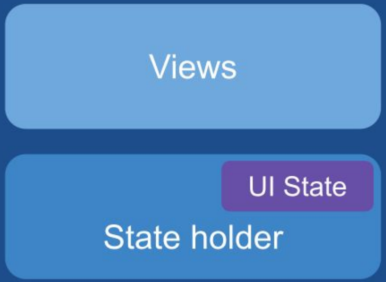
State /steɪt/ all the info needed to (re)build your UI

Types:

- Ephemeral state => setState() + StatefulWidget
- App state => state management technique



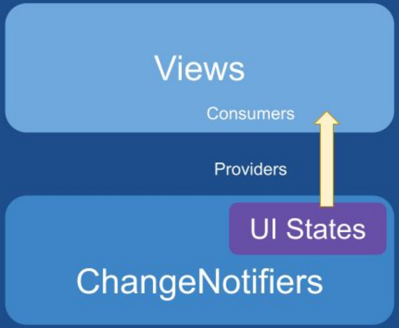
Ephemeral state



Application state

Simple State Management

- Only one officially documented (source)
- Riverpod+ChangeNotifier
- ChangeNotifier encapsulates the app state, and notifies about its changes
- Riverpod provides ChangeNotifier to the UI
- UI consumes changes with Riverpod



Riverpod

A Reactive Caching and Data-binding Framework

(source)

Architecture

ἀρχιτέκτων

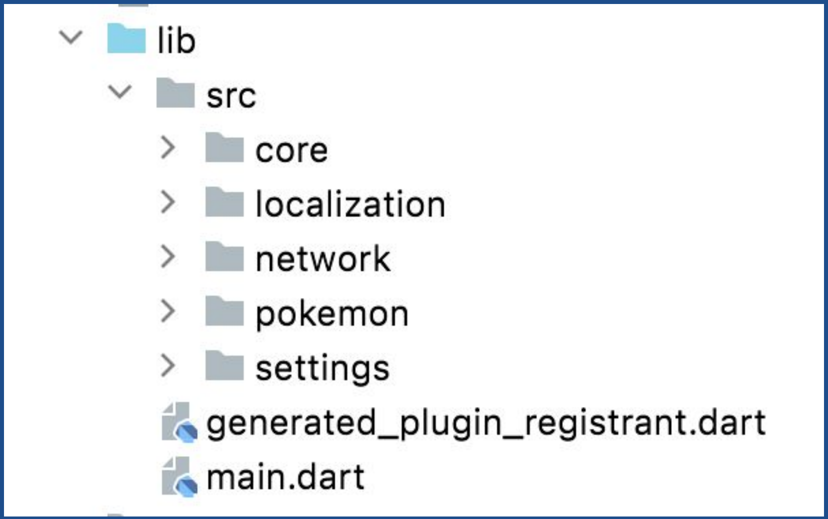
Architecture /ˈɑːkɪtɛktʃə/ (from arkhitéktōn, “chief builder”) the structure and design of a system

Purpose: maintainable, robust, scalable, testable

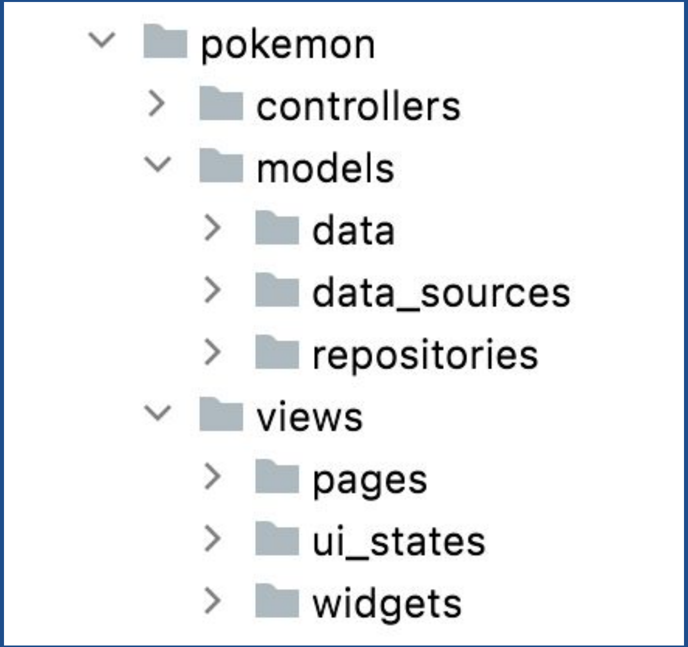
Principles:

- Separation of concerns
- Drive UI from (persistent) data models
- Single source of truth (SSOT)
- Unidirectional data flow (UDF)

Feature folders



```
lib
  src
    core
    localization
    network
    pokemon
    settings
    generated_plugin_registrant.dart
    main.dart
```



```
pokemon
  controllers
  models
    data
    data_sources
    repositories
  views
    pages
    ui_states
    widgets
```

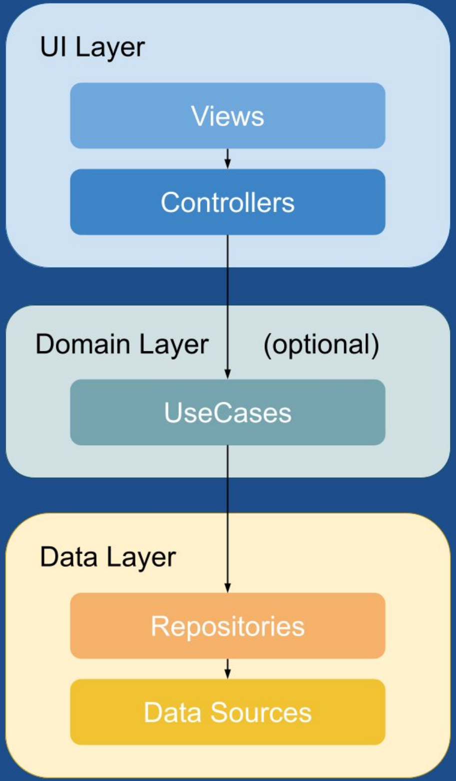
Rule of thumb: removing a feature should be as simple as removing the folder

Flutter 2.5 template

- flutter create -t skeleton flutter_arch_comp
- SettingsController with ChangeNotifier

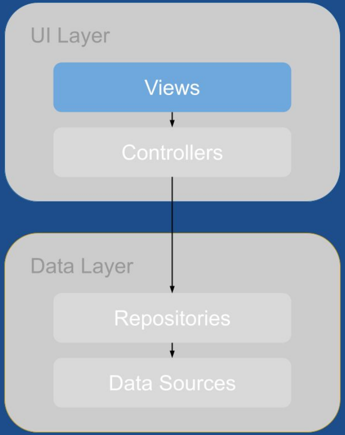
➔ Model View Controller (MVC)

MVC


UI Layer
Views
Controllers
Domain Layer
(optional)
UseCases
Data Layer
Repositories
Data Sources

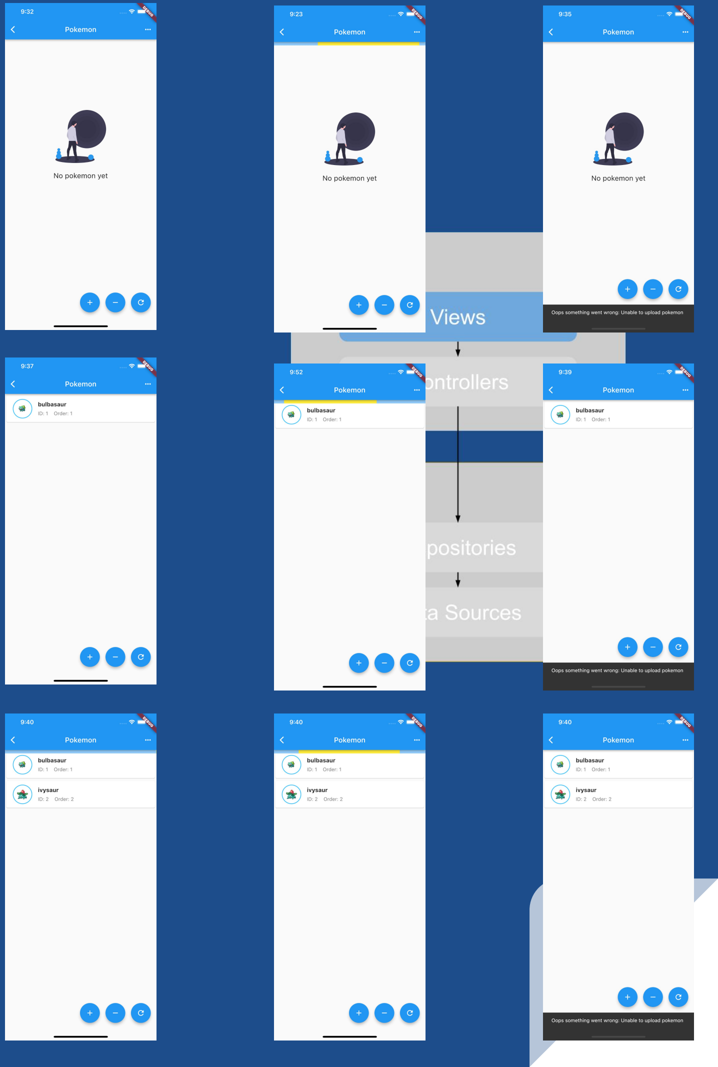
Views

- Display data on screen
- Capture user's interaction
- Visual representation of app state from data layer
- StatelessWidget (app state)
- StatefulWidget (ephemeral state)
- build() as fast as possible, no logic



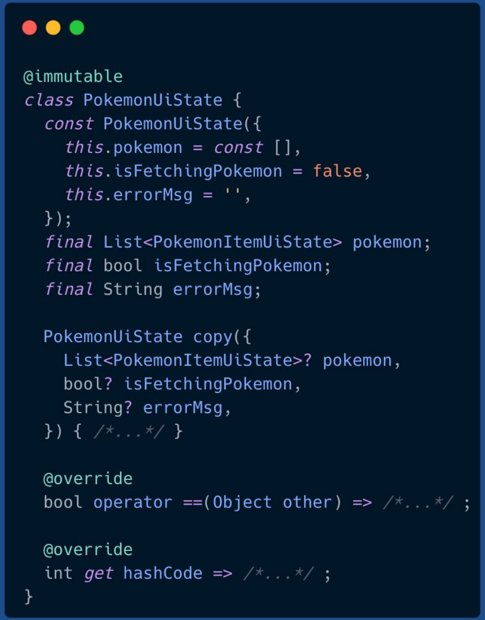
Views

- From imperative to declarative
- Rethink your UI mental model -> movie
- Finite state machine
- UI = f(state)

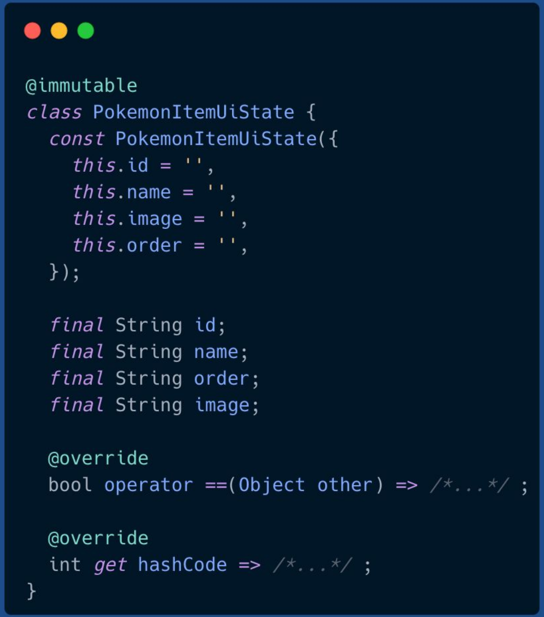


UI States

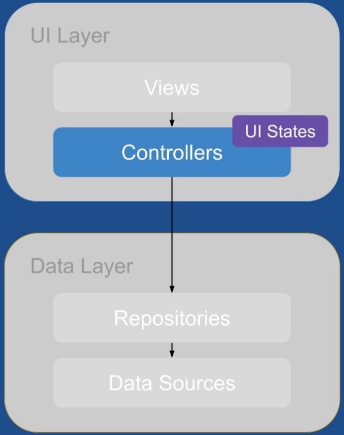
- Immutable
- Singular UIState class



```
@immutable
class PokemonUiState {
  const PokemonUiState({
    this.pokemon = const [],
    this.isFetchingPokemon = false,
    this.errorMsg = '',
  });
  final List<PokemonItemUiState> pokemon;
  final bool isFetchingPokemon;
  final String errorMsg;

  PokemonUiState copy({
    List<PokemonItemUiState>? pokemon,
    bool? isFetchingPokemon,
    String? errorMsg,
  }) { /*...*/ }

  @override
  bool operator ==(Object other) => /*...*/ ;

  @override
  int get hashCode => /*...*/ ;
}
```

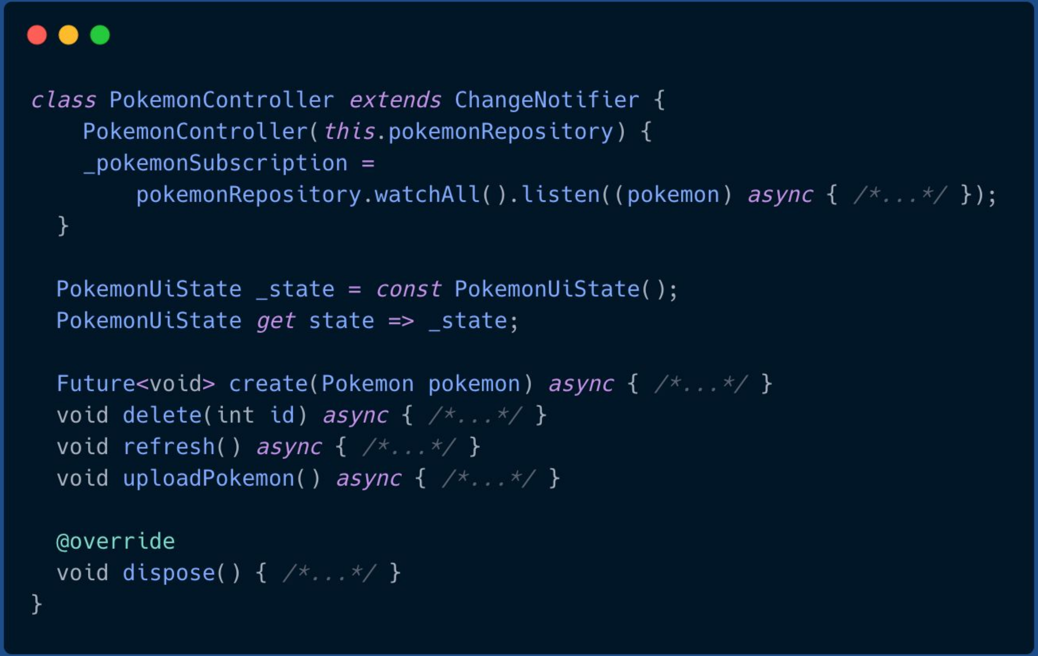


```
@immutable
class PokemonItemUiState {
  const PokemonItemUiState({
    this.id = '',
    this.name = '',
    this.image = '',
    this.order = '',
  });

  final String id;
  final String name;
  final String order;
  final String image;

  @override
  bool operator ==(Object other) => /*...*/ ;

  @override
  int get hashCode => /*...*/ ;
}
```



Controllers

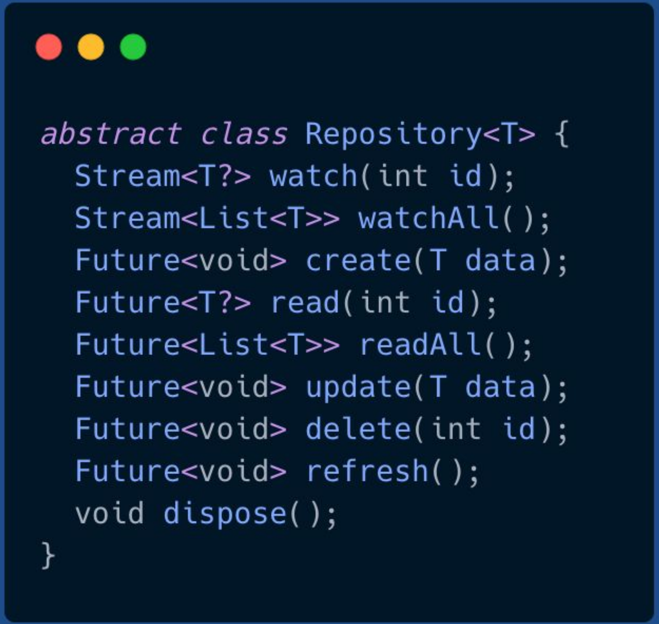
- Handle user's interaction
- Hold the UI state
- Expose the UI state for consumption
- Manipulate the UI state listening to data models



```dart
class PokemonController extends ChangeNotifier {
    PokemonController(this.pokemonRepository) {
    _pokemonSubscription =
        pokemonRepository.watchAll().listen((pokemon) async { /*...*/ });
  }

  PokemonUiState _state = const PokemonUiState();
  PokemonUiState get state => _state;

  Future<void> create(Pokemon pokemon) async { /*...*/ }
  void delete(int id) async { /*...*/ }
  void refresh() async { /*...*/ }
  void uploadPokemon() async { /*...*/ }

  @override
  void dispose() { /*...*/ }
}
```

Repositories

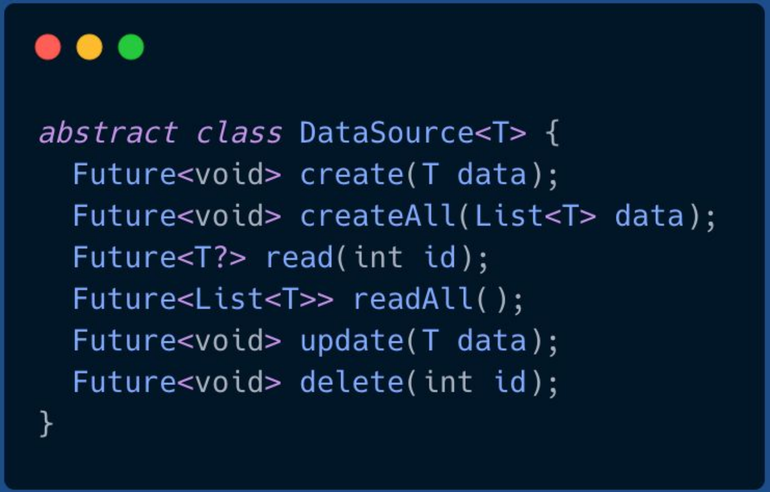
- Orchestrate between data sources (persistent model, web service, cache)
- One-shot CRUD operations, data changes over time
- Expose immutable data (trimmed down)
- Single source of truth (per repo)
- Business logic



```
abstract class Repository<T> {
  Stream<T?> watch(int id);
  Stream<List<T>> watchAll();
  Future<void> create(T data);
  Future<T?> read(int id);
  Future<List<T>> readAll();
  Future<void> update(T data);
  Future<void> delete(int id);
  Future<void> refresh();
  void dispose();
}
```

Data Sources

- One per source of data (db, network, file, shared prefs, etc)
- One-shot CRUD operations
- Accessed only by repositories

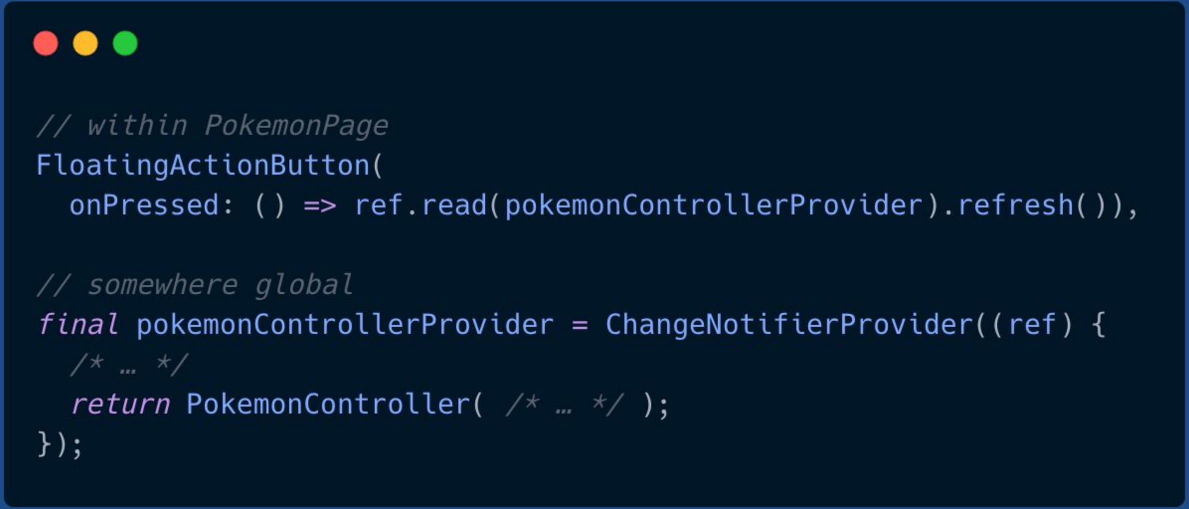


```
abstract class DataSource<T> {
  Future<void> create(T data);
  Future<void> createAll(List<T> data);
  Future<T?> read(int id);
  Future<List<T>> readAll();
  Future<void> update(T data);
  Future<void> delete(int id);
}
```

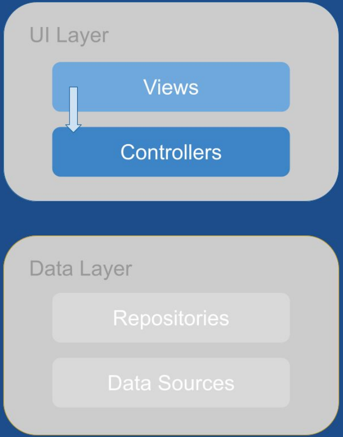
Interaction

Views => Controllers

- UI sends events upon user interaction

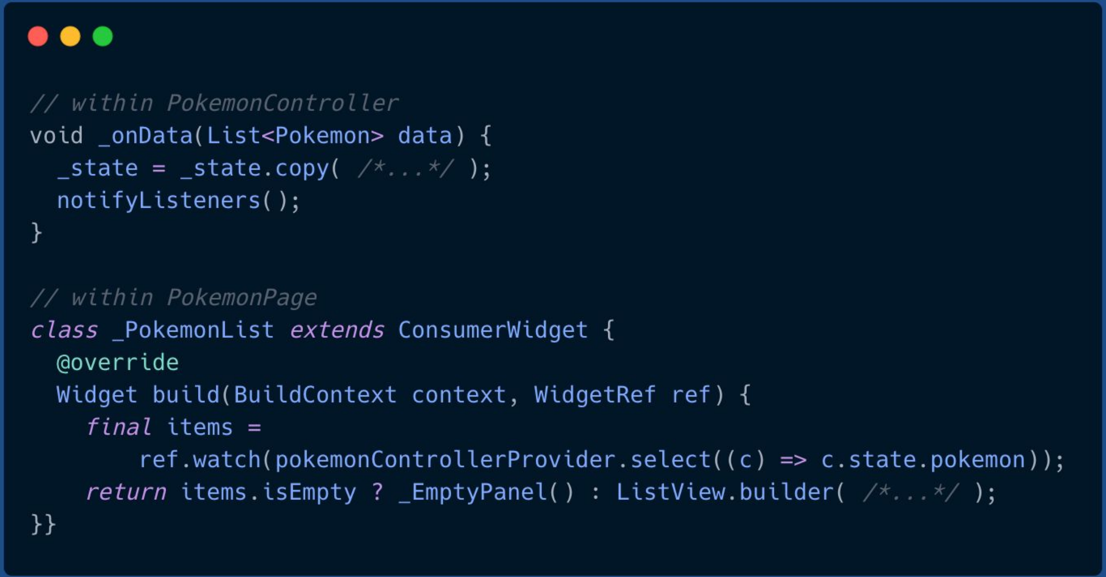


```
// within PokemonPage
FloatingActionButton(
  onPressed: () => ref.read(pokemonControllerProvider).refresh()),

// somewhere global
final pokemonControllerProvider = ChangeNotifierProvider((ref) {
  /* … */
  return PokemonController( /* … */ );
});
```

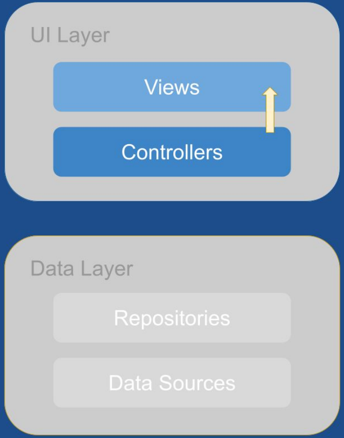


Views <= Controllers

- Controller dictates UI when to rebuild
- UI = f(state)

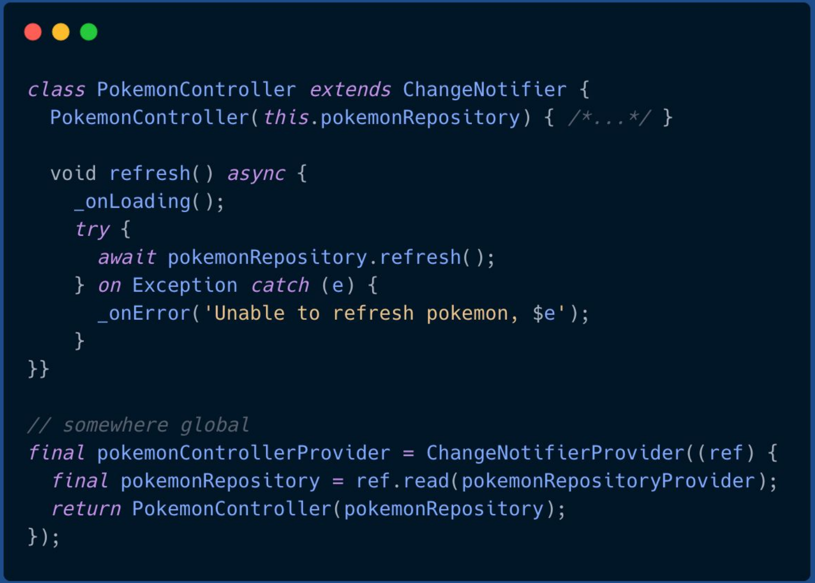


```
// within PokemonController
void _onData(List<Pokemon> data) {
  _state = _state.copy( /*...*/ );
  notifyListeners();
}

// within PokemonPage
class _PokemonList extends ConsumerWidget {
  @override
  Widget build(BuildContext context, WidgetRef ref) {
    final items =
        ref.watch(pokemonControllerProvider.select((c) => c.state.pokemon));
    return items.isEmpty ? _EmptyPanel() : ListView.builder( /*...*/ );
}}
```

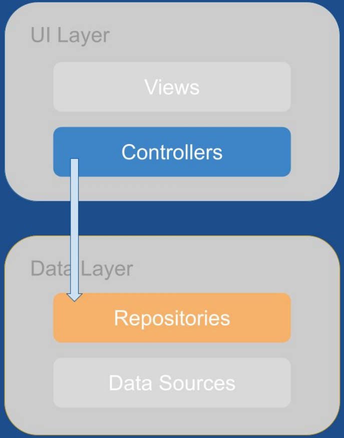


Controllers => Models

- Controller send events to change the data model
- Dependency Injection

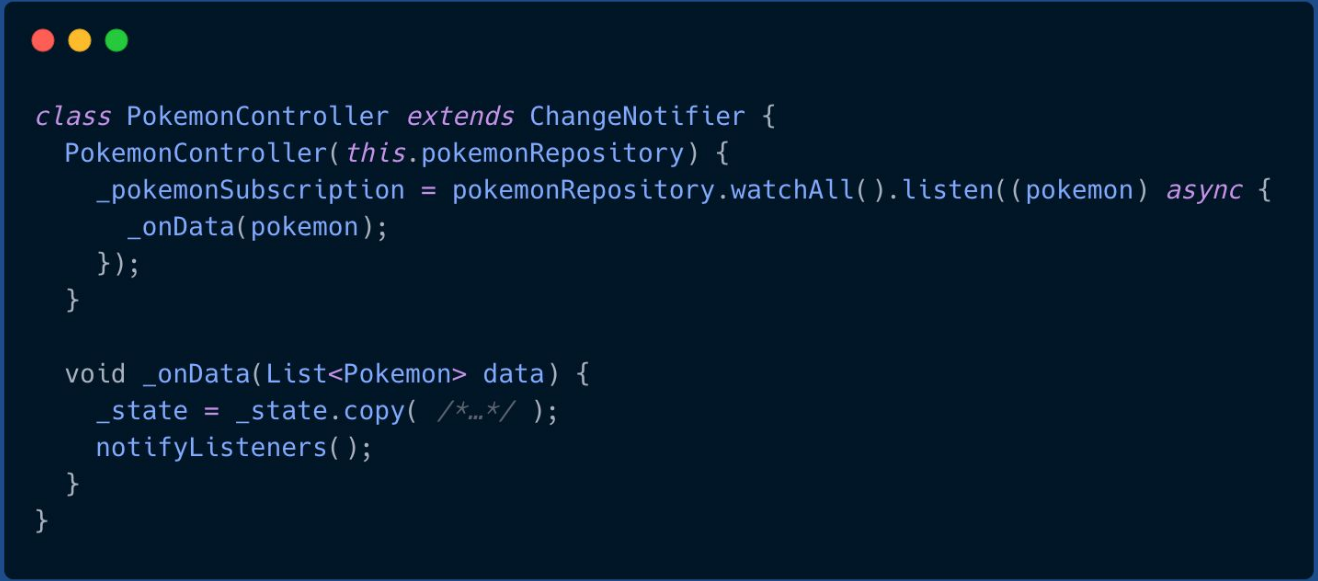


```dart
class PokemonController extends ChangeNotifier {
  PokemonController(this.pokemonRepository) { /*...*/ }

  void refresh() async {
    _onLoading();
    try {
      await pokemonRepository.refresh();
    } on Exception catch (e) {
      _onError('Unable to refresh pokemon, $e');
    }
}}

// somewhere global
final pokemonControllerProvider = ChangeNotifierProvider((ref) {
  final pokemonRepository = ref.read(pokemonRepositoryProvider);
  return PokemonController(pokemonRepository);
});
```

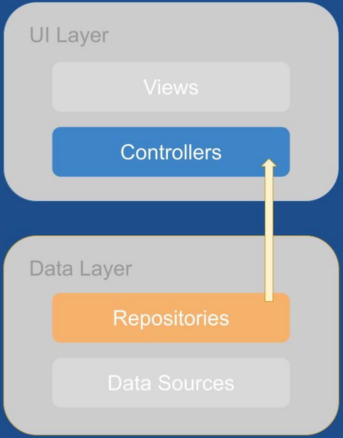


Controllers <= Models

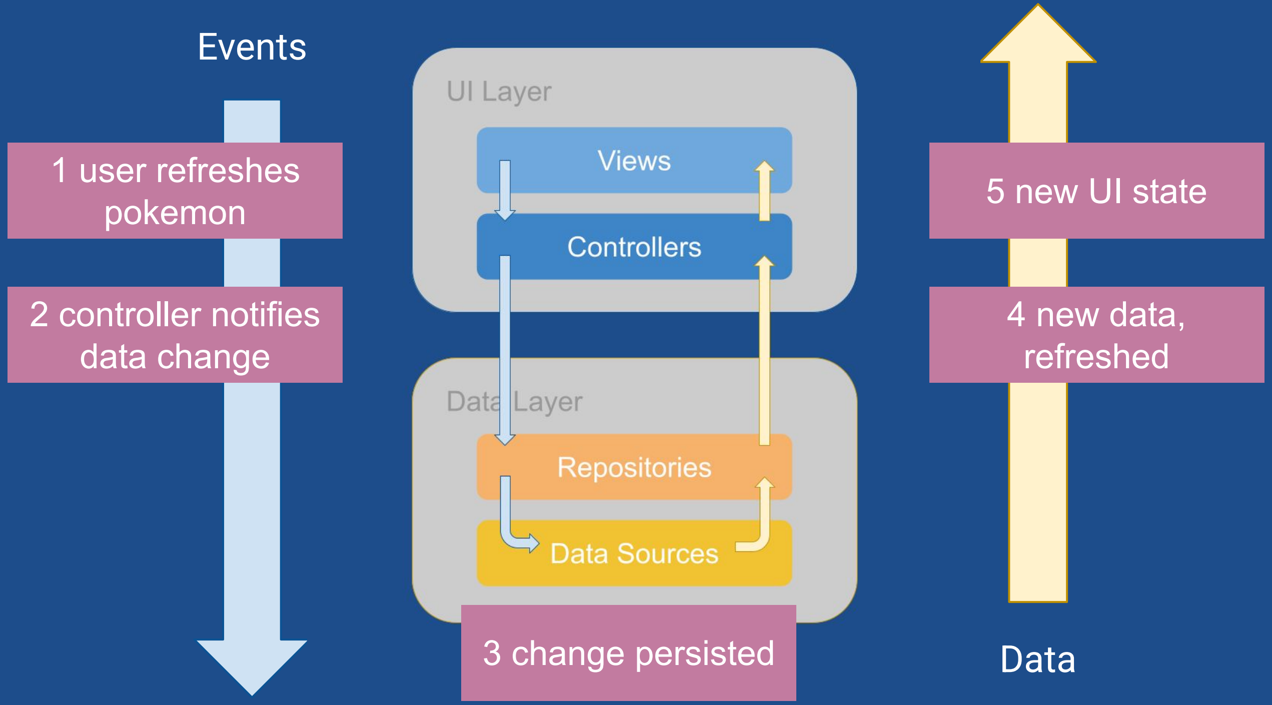
- Controller reacts to changes in the data model



```
class PokemonController extends ChangeNotifier {
  PokemonController(this.pokemonRepository) {
    _pokemonSubscription = pokemonRepository.watchAll().listen((pokemon) async {
      _onData(pokemon);
    });
  }

  void _onData(List<Pokemon> data) {
    _state = _state.copy( /*…*/ );
    notifyListeners();
  }
}
```

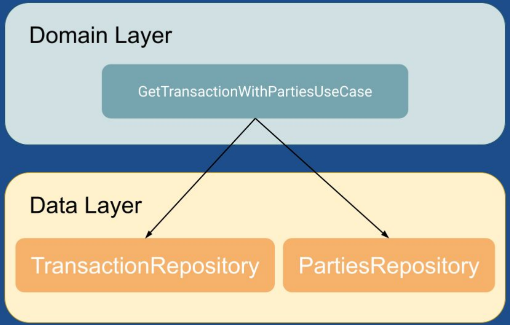


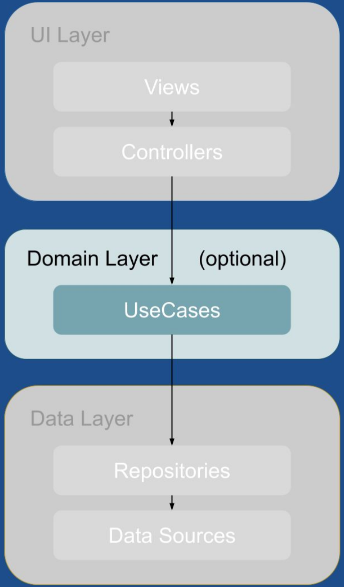
UDF


Events
1 user refreshes pokemon
2 controller notifies data change
UI Layer
Views
Controllers
Data Layer
Repositories
Data Sources
3 change persisted
5 new UI state
4 new data, refreshed
Data

Optional: Domain layer

- Reusable business logic
- Complex business logic
- UseCases ->
  - formatDateUseCase
  - LogOutUserUseCase
  - GetTransactionWithPartiesUseCase





Outro

Conclusions

- Many different architectures
- Define the metrics to compare them
- Simplicity
- Best -> ‘it depends’

➔ Code as a reflection of your mental model

Thank you!