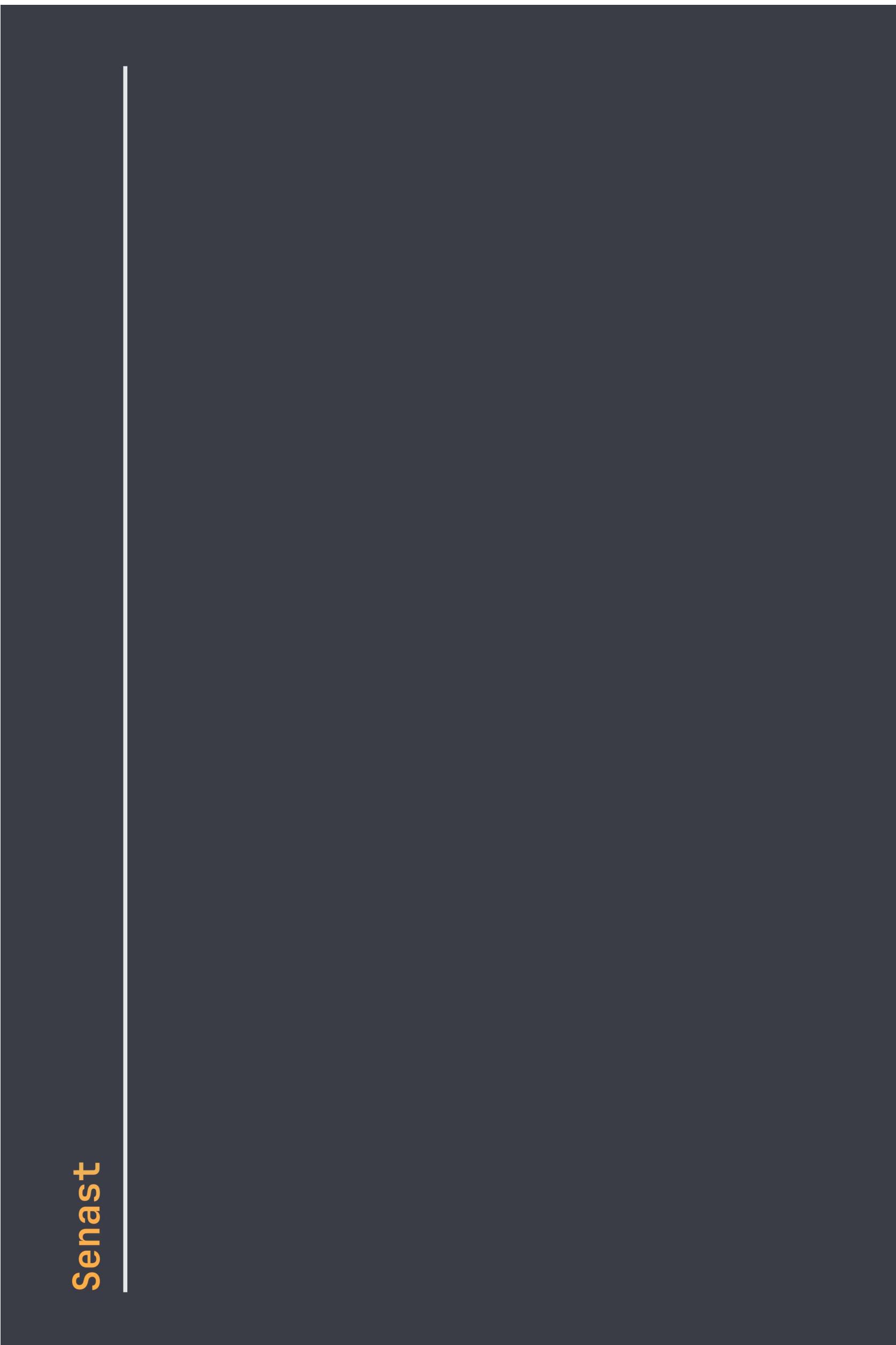


13:56:22

Hybridutveckling med Flutter





1. asynkrona anrop

Senast

1. asynkrona anrop
2. try/catch + felhantering

Senast

1. asynkrona anrop
2. try/catch + felhantering
3. använda extension methods

Senast

1. asynkrona anrop
2. try/catch + felhantering
3. använda extension methods
4. streams + sockets

Senast

1. asynkrona anrop
2. try/catch + felhantering
3. använda extension methods
4. streams + sockets
5. backend - programmering

Senast

1. asynkrona anrop
2. try/catch + felhantering
3. använda extension methods
4. streams + sockets
5. backend-programmering
6. tdd + testskrivning

Senast

1. asynkrona anrop
2. try/catch + felhantering
3. använda extension methods
4. streams + sockets
5. backend-programmering
6. tdd + testskrivning
7. felsökning + debugging

Handling

Handledning

1. Blandad handledning, frågor + svar + demo på begäran

Dagens agenda

Dagens agenda

1. Ny uppgift. Inlämning nästa måndag.

Dagens agenda

1. Ny uppgift. Inlämning nästa måndag.
2. Nya koncept kopplade till uppgiften

Dagens agenda

1. Ny uppgift. Inlämning nästa måndag.
2. Nya koncept kopplade till uppgiften

Nytt uppdrag: Inlämning nästa måndag.

Nytt uppgift: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.

Nytt uppgift: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.
2. Du tar över efter Arkitekt Perra på Clique King-projektet.

Nyttt uppgift: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.
 2. Du tar över efter Arkitekt Perra på Clique King-projektet.
- 3. Clique King:** En plattform där användare blir en del av "cliques" och strävar efter att bli kung.

Nytt uppdrag: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.
2. Du tar över efter Arkitekt Perra på Clique King-projektet.
3. **Clique King:** En plattform där användare blir en del av "cliques" och strävar efter att bli kung.
 - Kodbasen är avancerad, byggd med **BLoC**-mönstret för Flutter och **Firebase**-backend.

Nyttt uppgift: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.
2. Du tar över efter Arkitekt Perra på Clique King-projektet.
3. **Clique King:** En plattform där användare blir en del av "cliques" och strävar efter att bli kung.
 - Kodbasen är avancerad, byggd med **BLoC**-mönstret för Flutter och **Firebase**-backend.
4. **Föregående uppgift:** Mycket mindre frihet. Mindre nytt att implementera

Nyttt uppgift: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.
2. Du tar över efter Arkitekt Perra på Clique King-projektet.
3. **Clique King:** En plattform där användare blir en del av "cliques" och strävar efter att bli kung.
 - Kodbasen är avancerad, byggd med **BLoC**-mönstret för Flutter och **Firebase**-backend.
4. Föregående uppgift: Mycket mindre frihet. Mindre nytt att implementera
5. **Denna uppgift:** Mycket mer frihet. Mer nytt att implementera

Nyttt uppgift: Inlämning nästa måndag.

1. CTO Petter gratulerar till befordran.
2. Du tar över efter Arkitekt Perra på Clique King-projektet.
3. **Clique King:** En plattform där användare blir en del av "cliques" och strävar efter att bli kung.
 - Kodbasen är avancerad, byggd med **BLoC**-mönstret för Flutter och **Firebase**-backend.
4. Föregående uppgift: Mycket mindre frihet. Mindre nytt att implementera
5. Denna uppgift: Mycket mer frihet. Mer nytt att implementera
6. Överlämning från **Arkitekt Perra**.

Överlämning från Arkitekt Perra

Överlämning från Arkitekt Perra

1. **Introduktion:** Läs först mailkorrespondens från CTO.

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - **Har exempel för paketet bloc**

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o **Har exempel för paketet `bloc_test`**

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för `Firebase` kommunikation med paketet `firedart`

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för **Firebase** kommunikation med paketet `firedart`
 - o Läser Environment Variables från `.env` med paketet `dotenv`

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för **Firebase** kommunikation med paketet `firedart`
 - o Läser Environment Variables från `.env` med paketet `dotenv`
 - o **Har TRE repositories du förväntas implementera**

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för **Firebase** kommunikation med paketet `firedart`
 - o Läser Environment Variables från `.env` med paketet `dotenv`
 - o Har **TRE** repositories du förväntas implementera
 - o Har **TRE blocs** du förväntas implementera

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för **Firebase** kommunikation med paketet `firedart`
 - o Läser Environment Variables från `.env` med paketet `dotenv`
 - o Har **TRE** repositories du förväntas implementera
 - o Har **TRE** blocs du förväntas implementera
 - **Dessa har States och Events definierade redan**

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för `Firebase` kommunikation med paketet `firedart`
 - o Läser Environment Variables från `.env` med paketet `dotenv`
 - o Har `TRE` repositories du förväntas implementera
 - o Har `TRE` blocs du förväntas implementera
 - Dessa har States och Events definierade redan
 - **Upp till dig att rätt sak händer för rätt event och att rätt state skapas**

Överlämning från Arkitekt Perra

1. Introduktion: Läs först mailkorrespondens från CTO.
2. Projektöversikt: ett dart-projekt
 - o Har exempel för paketet `bloc`
 - o Har exempel för paketet `bloc_test`
 - o Har exempel för **Firebase** kommunikation med paketet `firedart`
 - o Läser Environment Variables från `.env` med paketet `dotenv`
 - o Har **TRE** repositories du förväntas implementera
 - o Har **TRE** blocs du förväntas implementera
 - Dessa har States och Events definierade redan
 - Upp till dig att hantera så att **rätt sak** händer för **rätt event** och att **rätt state** skapas
 - o Finns en del TODO:s men inte tillräckligt

Kravspecifikation för Clique King

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.

Kravspecifikation för Claque King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. **Se, skapa och delta i cliques som inloggad användare.**

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.
5. Alla clique-medlemmar får ett startvärde på 0.

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.
5. Alla clique-medlemmar får ett startvärde på 0.
6. Värdeökning genom "actions" inom en clique.

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.
5. Alla clique-medlemmar får ett startvärde på 0.
6. Värdeökning genom "actions" inom en clique.
7. Medlem med högst värde blir "Clique King".

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.
5. Alla clique-medlemmar får ett startvärde på 0.
6. Värdeökning genom "actions" inom en clique.
7. Medlem med högst värde blir "Clique King".
8. "Actions" kan vara valfria, exempelvis ett knappklick. Mer avancerat exempel: stegräkning.

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.
5. Alla clique-medlemmar får ett startvärde på 0.
6. Värdeökning genom "actions" inom en clique.
7. Medlem med högst värde blir "Clique King".
8. "Actions" kan vara valfria, exempelvis ett knappklick. Mer avancerat exempel: stegräkning.
9. Alla "actions" representeras som Events i bloc.

Kravspecifikation för Clique King

1. Registrering & inloggning via Firebase Authentication.
2. Lagring av användardata i Firebase Firestore.
3. Se, skapa och delta i cliques som inloggad användare.
4. Se medlemsinformation när en specifik clique väljs.
5. Alla clique-medlemmar får ett startvärde på 0.
6. Värdeökning genom "actions" inom en clique.
7. Medlem med högst värde blir "Clique King".
8. "Actions" kan vara valfria, exempelvis ett knappklick. Mer avancerat exempel: stegräkning.
9. Alla "actions" representeras som Events i bloc.
- 10. Data för vyer representeras som States i bloc.**

Dagens agenda

1. Ny uppgift. Inlämning nästa måndag.
2. Nya koncept kopplade till uppgiften

Nya koncept kopplade till uppgifter

Nya koncept kopplade till uppgifter

1. event-driven-architecture

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. **firebase authentication (stödjer flera login providers)**

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. firebase authentication (stödjer flera login providers)
7. environment variables

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. firebase authentication (stödjer flera login providers)
7. environment variables
8. Du får också öva på att navigera, modifiera och komplettera kod i ett existerande projekt

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. firebase authentication (stödjer flera login providers)
7. environment variables
8. Du får också öva på att navigera, modifiera och komplettera kod i ett existerande projekt
 - Representativt för riktigt arbete

event-driven-architecture

event-driven-architecture

1. Baserad på att fångga och svara på häändelser. Ger ett dynamiskt sätt att hantera interaktioner och dataflöde.

event-driven-architecture

1. Baserad på att fångga och svara på häändelser. Ger ett dynamiskt sätt att hantera interaktioner och dataflöde.
2. Förhättar skalbarhet och underlättar integration mellan olika delar av systemet.

event-driven-architecture

1. Baserad på att fångar och svara på häändelser. Ger ett dynamiskt sätt att hantera interaktioner och dataflöde.
2. Förbättrar skalbarhet och underlättar integration mellan olika delar av systemet.
3. **BLoC-mönstret** använder denna arkitektur: UI komponenter skickar events till BLoC och BLoC sänder tillbaka states till UI.

BLoC-mönstret (Business Logic Component)

BLoC-mönstret (Business Logic Component)

- 1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsupdelning.**

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessas händelser producerar BLoC nya och olika states.

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessas händelser producerar BLoC nya och olika states.
3. Ger en reaktiv lösning för state-hantering.

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessas händelser producerar BLoC nya och olika states.
3. Ger en reaktiv lösning för state-hantering.
4. Lätt att testa och underhålla.

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessas händelser producerar BLoC nya och olika states.
3. Ger en reaktiv lösning för state-hantering.
4. Lätt att testa och underhålla.
5. Gör det möjligt för utvecklare att uttrycka en applikations samtliga tillstånd och definiera hur tillståndsförändringar kan och får ske.

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessas händelser producerar BLoC nya och olika states.
3. Ger en reaktiv lösning för state-hantering.
4. Lätt att testa och underhålla.
5. Gör det möjligt för utvecklare att uttrycka en applikations samtliga tillstånd och definiera hur tillståndsförändringar kan och får ske.
6. Framtagen för att underlätta samarbete i stora utvecklingsteam.

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessas händelser producerar BLoC nya och olika states.
3. Ger en reaktiv lösning för state-hantering.
4. Lätt att testa och underhålla.
5. Gör det möjligt för utvecklare att uttrycka en applikations samtliga tillstånd och definiera hur tillståndsförändringar kan och får ske.
6. Framtagen för att underlätta samarbete i stora utvecklingsteam.
7. Den **första** riktigt väladoptera Lösningen för att hantera state i Flutter.

BLoC-mönstret (Business Logic Component)

1. Separerar presentationen från affärslogiken. Ger en ren arkitektur med tydlig ansvarsuppdelning.
2. UI-komponenter skickar events till BLoC. Baserat på dessa händelser producerar BLoC nya och olika states.
3. Ger en reaktiv lösning för state-hantering.
4. Lätt att testa och underhålla.
5. Gör det möjligt för utvecklare att uttrycka en applikations samtliga tillstånd och definiera hur tillståndsförändringar kan och får ske.
6. Framtagen för att underlätta samarbete i stora utvecklingsteam.
7. Den **första** riktigt väladopterade lösningen för att hantera state i Flutter.
 - **Bland annat** därför den också är mest använd.

BLOC - Exempl

```
1 class CounterState {  
2   final int value;  
3  
4   CounterState(this.value);  
5 }
```

BLoC - Exemplar

```
1 class CounterState {  
2   final int value;  
3  
4   CounterState(this.value);  
5 }  
6  
7 sealed class CounterEvent {}  
8  
9 final class CounterIncrement extends CounterEvent {}
```

BLOC - Exempl

```
1 class CounterState {
2   final int value;
3
4   CounterState(this.value);
5 }
6
7 sealed class CounterEvent {}
8
9 final class CounterIncrement extends CounterEvent {}
10
11 class CounterBloc extends Bloc<CounterEvent, CounterState> {
12   CounterBloc() : super(CounterState(0)) {
13     on<CounterEvent>((event, emit) {
14       switch (event) {
15         case CounterIncrement():
16           emit(CounterState(state.value + 1));
17         }
18       });
19     }
20 }
```

BLOC - Exempl

```
1 class CounterState {  
2   final int value;  
3  
4   CounterState(this.value);  
5 }  
6  
7 sealed class CounterEvent {}  
8  
9 final class CounterIncrement extends CounterEvent {}  
10  
11 class CounterBloc extends Bloc<CounterEvent, CounterState> { // Bloc<Event, State>  
12   CounterBloc() : super(CounterState(0)) {  
13     on<CounterEvent>((event, emit) {  
14       switch (event) {  
15         case CounterIncrement():  
16           emit(CounterState(state.value + 1));  
17         }  
18       });  
19     }  
20   }
```

BLOC - Exempl

```
1 class CounterState {
2   final int value;
3
4   CounterState(this.value);
5 }
6
7 sealed class CounterEvent {}
8
9 final class CounterIncrement extends CounterEvent {}
10
11 class CounterBloc extends Bloc<CounterEvent, CounterState> {
12   CounterBloc() : super(CounterState(0)) { // define initial state of bloc with super constructor call
13     on<CounterEvent>((event, emit) {
14       switch (event) {
15         case CounterIncrement():
16           emit(CounterState(state.value + 1));
17         }
18       });
19     }
20   }
```

BLOC - Exempl

```
1 class CounterState {  
2   final int value;  
3  
4   CounterState(this.value);  
5 }  
6  
7 sealed class CounterEvent {}  
8  
9 final class CounterIncrement extends CounterEvent {}  
10  
11 class CounterBloc extends Bloc<CounterEvent, CounterState> {  
12   CounterBloc() : super(CounterState(0)) {  
13     on<CounterEvent>((event, emit) { // define event handler, read the event and use emit callback to emit new state  
14       switch (event) {  
15         case CounterIncrement():  
16           emit(CounterState(state.value + 1));  
17         }  
18       }  
19     }  
20   }
```

BLOC - Exempl

```
1 class CounterState {
2   final int value;
3
4   CounterState(this.value);
5 }
6
7 sealed class CounterEvent {}
8
9 final class CounterIncrement extends CounterEvent {}
10
11 class CounterBloc extends Bloc<CounterEvent, CounterState> {
12   CounterBloc() : super(CounterState(0)) {
13     on<CounterEvent>((event, emit) {
14       switch (event) { // for the different events, handle event and emit new state
15         case CounterIncrement():
16           emit(CounterState(state.value + 1));
17         }
18       });
19     }
20   }
```

Testing av Bloc (Business Logic Component)

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstestning av affärslogik utan beroenden av UI eller externa tjänster.

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstesting av affärslogik utan beroenden av UI eller externa tjänster.
2. **bloc_test** biblioteket erbjuder verktyg för att förenkla **testprocessen**.

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstesting av affärslogik utan beroenden av UI eller externa tjänster.
2. `bloc_test` biblioteket erbjuder verktyg för att förenkla testprocessen.
3. Tester kan replikera events och förvänta sig specifika states som svar.

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstesting av affärslogik utan beroenden av UI eller externa tjänster.
2. `bloc_test` biblioteket erbjuder verktyg för att förenkla testprocessen.
3. Tester kan replikera events och förvänta sig specifika states som svar.
- 4.Verifiera att Bloc reagerar korrekt på varje given händelse.

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstestning av affärslogik utan beroenden av UI eller externa tjänster.
2. `bloc_test` biblioteket erbjuder verktyg för att förenkla testprocessen.
3. Tester kan replikera events och förvänta sig specifika states som svar.
- 4.Verifiera att Bloc reagerar korrekt på varje given händelse.
5. Ger säkerhet när man utvecklar eller refaktorisar

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstestning av affärslogik utan beroenden av UI eller externa tjänster.
 2. `bloc_test` biblioteket erbjuder verktyg för att förenkla testprocessen.
 3. Tester kan replikera events och förvänta sig specifika states som svar.
 - 4.Verifiera att Bloc reagerar korrekt på varje given händelse.
 5. Ger säkerhet när man utvecklar eller refaktoriseringar
- 6. Förhindrar applikationens robusthet genom att upptäcka och förhindra buggar.**

Testing av Bloc (Business Logic Component)

1. Möjliggör enhetstesting av affärslogik utan beroenden av UI eller externa tjänster.
2. `bloc_test` biblioteket erbjuder verktyg för att förenkla testprocessen.
3. Tester kan replikera events och förvänta sig specifika states som svar.
- 4.Verifiera att Bloc reagerar korrekt på varje given händelse.
5. Ger säkerhet när man utvecklar eller refaktoriseringar
6. Förbättrar applikationens robusthet genom att upptäcka och förhindra buggar.
7. Möjliggör testdriven utveckling (TDD) inom Flutter-ekosystemet.

BLOC test - exempl

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(),
5     act: (bloc) {},
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9});
```

BLOC test - exemplu

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created', // create a new test and name it
4     build: () => CounterBloc(),
5     act: (bloc) {},
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9 });
```

BLOC test - exempl

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(), // build should construct and return the bloc being tested
5     act: (bloc) {},
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9 });
```

BLOC test - exemplar

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(),
5     act: (bloc) => bloc,
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9 });
```

BLOC test - exempl

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(),
5     act: (bloc) {},
6     expect: () => [], // array of emitted states expected after act runs. None since no emitted state yet,
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9 });
```

BLOC test - exemple

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(),
5     act: (bloc) {},
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0), // asserts can be ran here to check the final state
8   );
9 });
```

BLOC test - exempl

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(),
5     act: (bloc) {},
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9
10  blocTest(
11    'Emits incremented states on CounterIncrement event',
12    build: () => CounterBloc(),
13    act: (bloc) {
14      bloc.add(CounterIncrement());
15      bloc.add(CounterIncrement());
16      bloc.add(CounterIncrement());
17    },
18    expect: () => [
19      CounterState(1),
20      CounterState(2),
21      CounterState(3),
22    ],
23  );
24});
```

BLOC test - exemple

```
1 group('CounterBloc Tests', () {
2   blocTest(
3     'Nothing emitted when created',
4     build: () => CounterBloc(),
5     act: (bloc) {},
6     expect: () => [],
7     verify: (bloc) => bloc.state == CounterState(0),
8   );
9
10  blocTest(
11    'Emits incremented states on CounterIncrement event',
12    build: () => CounterBloc(),
13    act: (bloc) {
14      bloc.add(CounterIncrement());
15      bloc.add(CounterIncrement());
16      bloc.add(CounterIncrement());
17    },
18    expect: () => [
19      CounterState(1),
20      CounterState(2),
21      CounterState(3),
22      CounterState(4),
23    ];
24  );
25}
```

Firebase - En molnbaserad BaaS platform

Firebase - En molnbaserad Baas plattform

1. Erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.

Firebase - En molnbaserad Baas plattform

1. Er erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.
2. **Autentisering, databaser, lagring, hosting och mycket mer.**

Firebase - En molnbaserad Baas plattform

1. Er erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.
 2. Autentisering, databaser, lagring, hosting och mycket mer.
- 3. Stöder realtidsapplikationer med Firestore - en realtids NoSQL-databas.**

Firebase - En molnbaserad **BaaS** plattform

1. Er erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.
2. Autentisering, databaser, lagring, hosting och mycket mer.
3. Stöder realtidsapplikationer med **Firestore** - en realtids NoSQL-databas.
4. Optimerad för utveckling av mobilappar med **SDKs** för **stora** plattformar som **iOS**, **Android** och **webb**.

Firebase - En molnbaserad Baas plattform

1. Erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.
2. Autentisering, databaser, lagring, hosting och mycket mer.
3. Stöder realtidsapplikationer med **Firestore** - en realtids NoSQL-databas.
4. Optimerad för utveckling av mobilappar med SDKs för stora plattformar som iOS, Android och webb.
5. Underlättar autentisering med inbyggda tjänster som stöder e-post, telefon och sociala medieinloggningar.

Firebase - En molnbaserad Baas plattform

1. Erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.
2. Autentisering, databaser, lagring, hosting och mycket mer.
3. Stöder realtidsapplikationer med **Firestore** - en realtids NoSQL-databas.
4. Optimerad för utveckling av mobilappar med SDKs för stora plattformar som iOS, Android och webb.
5. Underlättar autentisering med inbyggda tjänster som stöder e-post, telefon och sociala medieinloggningar.
6. Säkerhetsfunktioner som hjälper till att skydda appdata och användarinformation.

Firebase - En molnbaserad Baas plattform

1. Erbjuder en mängd utvecklingsverktyg och tjänster för att bygga, förbättra och skala appar.
2. Autentisering, databaser, lagring, hosting och mycket mer.
3. Stöder realtidsapplikationer med **Firestore** - en realtids NoSQL-databas.
4. Optimerad för utveckling av mobilappar med SDKs för stora plattformar som iOS, Android och webb.
5. Underlättar autentisering med inbyggda tjänster som stöder e-post, telefon och sociala medieinloggningar.
6. Säkerhetsfunktioner som hjälper till att skydda appdata och användarinformation.
 - T.ex. säkerhetsregler för **Firebase Firestore och Storage**.

Firebase Firestore - En realtids NOSQL-database

Firebase Firestore - En **realtidss NOSQL**-databas

1. Erbjuder flexibel, skalbar och **realtidsdatabashantering** för mobil- och webbapplikationer.

Firebase Firestore - En **realtidss** NOSQL-databas

1. Erbjuder flexibel, skalbar och realtidssdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.

Firebase Firestore - En **realtidss NOSQL**-databas

1. Erbjuder flexibel, skalbar och realtidssdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.
3. Varje dokument kan innehålla komplexa datastrukturer, inklusive nästlade samlingar.

Firebase Firestore - En **realtidss** NoSQL-databas

1. Erbjuder flexibel, skalbar och realtidssdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.
3. Varje dokument kan innehålla komplexa datastrukturer, inklusive nästlade samlingar.
4. **Stöd för offline data:** Appen kan fortfarande läsa och skriva data även när den är offline.

Firebase Firestore - En **realtidss** NOSQL-databas

1. Erbjuder flexibel, skalbar och realtidssdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.
3. Varje dokument kan innehålla komplexa datastrukturer, inklusive nästlade samlingar.
4. Stöd för offline data: Appen kan fortfarande läsa och skriva data även när den är offline.
5. **Säkerhet på dokumentnivå med hjälp av Cloud Firestore säkerhetsregler.**

Firebase Firestore - En **realtidss** NOSQL-databas

1. Erbjuder flexibel, skalbar och realtidssdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.
3. Varje dokument kan innehålla komplexa datastrukturer, inklusive nästlade samlingar.
4. Stöd för offline data: Appen kan fortfarande läsa och skriva data även när den är offline.
5. Säkerhet på dokumentnivå med hjälp av Cloud Firestore säkerhetsregler.
6. Starka query-funktioner, vilket gör det möjligt att hämta och filtrera data med precision.

Firebase Firestore - En **realtidss** NOSQL-databas

1. Erbjuder flexibel, skalbar och realtidssdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.
3. Varje dokument kan innehålla komplexa datastrukturer, inklusive nästlade samlingar.
4. Stöd för offline data: Appen kan fortfarande läsa och skriva data även när den är offline.
5. Säkerhet på dokumentnivå med hjälp av Cloud Firestore säkerhetsregler.
6. Starka query-funktioner, vilket gör det möjligt att hämta och filtrera data med precision.
7. Inbyggd skalbarhet som automatiskt skalar upp och ner baserat på behov.

Firebase Firestore - En **realtids** NOSQL-databas

1. Erbjuder flexibel, skalbar och realtidsdatabashantering för mobil- och webbapplikationer.
2. Data lagras i dokument som är organiserade i samlingar.
3. Varje dokument kan innehålla komplexa datastrukturer, inklusive nästlade samlingar.
4. Stöd för offline data: Appen kan fortfarande läsa och skriva data även när den är offline.
5. Säkerhet på dokumentnivå med hjälp av Cloud Firestore säkerhetsregler.
6. Starka query-funktioner, vilket gör det möjligt att hämta och filtrera data med precision.
7. Inbyggd skalbarhet som automatiskt skalar upp och ner baserat på behov.
 - Integreras sömlöst med andra Firebase-produkter och Google Cloud-tjänster.

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exempel (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Firestore - exemplo (firedart)

```
1 Firestore.initializeApp(projectId); // required
2
3 Firestore.instance.collection("users").stream.listen((event) {
4   print(event); // any time anything in users collection changes, get the entire collection as an event
5 });
6
7 final user = User(name:"will");
8 final document = await Firestore.instance.collection("users").add(user.toMap());
9
10 Firestore.instance.collection("users").document(document.id).stream.listen(
11   (event) => print(event) // any time user changes at db, get event notified
12 );
13
14 final temp = user.copyWith(name:"kasper");
15 await Firestore.instance.collection("users").document(document.id).update(temp); // listener above gets kasper
16
17 await Firestore.instance.collection("users").document(document.id).delete(); // listener above gets null
```

Firebase Authentication - Enkel och säker inloggning

Firebase Authentication - Enkel och säker inloggning

1. Erjuder enkla, snabba och säkra metoder för att lägga till användarautentisering i applikationer.

Firebase Authentication - Enkel och säker inloggning

1. Erjuder enkla, snabba och säkra metoder för att lägga till användarautentisering i applikationer.
2. Stöder flera inloggningssmetoder: e-post och lösenord, telefonnummer, samt tredjejärtsleverantörer som Google, Facebook, Twitter och flera andra.

Firebase Authentication - Enkel och säker inloggning

1. Erjuder enkla, snabba och säkra metoder för att lägga till användarautentisering i applikationer.
2. Stöder flera inloggningssmetoder: e-post och lösenord, telefonnummer, samt tredjejärtsleverantörer som Google, Facebook, Twitter och flera andra.
3. Integrerad med Firebase ekosystemet, vilket gör det lätt att synkronisera användarinformation med andra Firebase-tjänster.

Firebase Authentication - Enkel och säker inloggning

1. Erjuder enkla, snabba och säkra metoder för att lägga till användarautentisering i applikationer.
2. Stöder flera inloggningssmetoder: e-post och lösenord, telefonnummer, samt tredjejärtsleverantörer som Google, Facebook, Twitter och flera andra.
3. Integrerad med Firebase ekosystemet, vilket gör det lätt att synkronisera användarinformation med andra Firebase-tjänster.
4. **Möjlighet att hantera användarsessioner, återställa lösenord och hantera användarrättigheter.**

Firebase Authentication - Enkel och säker inloggning

1. Erjuder enkla, snabba och säkra metoder för att lägga till användarautentisering i applikationer.
2. Stöder flera inloggningssmetoder: e-post och lösenord, telefonnummer, samt tredjejärtsleverantörer som Google, Facebook, Twitter och flera andra.
3. Integrerad med Firebase ekosystemet, vilket gör det lätt att synkronisera användarinformation med andra Firebase-tjänster.
4. Möjlighet att hantera användarsessioner, återställa lösenord och hantera användarrättigheter.
5. Förbättrad säkerhet med flerfaktorautentisering och skydd mot vanliga säkerhetsshot som bruteforce-attacker.

Firebase Authentication - Enkel och säker inloggning

1. Erjuder enkla, snabba och säkra metoder för att lägga till användarautentisering i applikationer.
2. Stöder flera inloggningssmetoder: e-post och lösenord, telefonnummer, samt tredjejärtsleverantörer som Google, Facebook, Twitter och flera andra.
3. Integrerad med Firebase ekosystemet, vilket gör det lätt att synkronisera användarinformation med andra Firebase-tjänster.
4. Möjlighet att hantera användarsessioner, återställa lösenord och hantera användarrättigheter.
5. Förbättrad säkerhet med flerfaktorautentisering och skydd mot vanliga säkerhetsshot som bruteforce-attacker.
6. Utbud av SDK:er för olika plattformar och språk, inklusive webb, iOS, Android, C++ och Dart.

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen((event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 });
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when singInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen(event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 }
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when singInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen((event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 });
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when signInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen(event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 }
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when singInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen((event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 });
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when singInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen((event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 });
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when singInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Firebase Authentication - exemple (firedart)

```
1 final path = Directory.current.path;
2 FirebaseAuth.initializeApp(apiKey, await HiveStore.create(path: path)); // Hive for keeping auth token
3 Firestore.initialize(projectId);
4
5 await FirebaseAuth.instance.signUp("tester@test.com", "test123"); // automatically signs in on success
6 await FirebaseAuth.instance.signIn("tester@lester.com", "test123"); // can also do this
7
8 FirebaseAuth.instance.signInState.listen(event) {
9   print(event); // any time the signinstate changes, get event, such as getting logged in/out
10 }
11
12 var user = await FirebaseAuth.instance.getUser(); // returns user when singInState=true
13
14 await FirebaseAuth.instance.deleteAccount(); // removes user from firebase authentication
```

Environment variables och .env-filer

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.
2. Ger en mekanism för att separera konfiguration från kod, vilket ökar säkerheten och flexibiliteten.

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.
2. Ger en mekanism för att separera konfiguration från kod, vilket ökar säkerheten och flexibiliteten.
3. **.env-filer används ofta för att lagra miljövariabler på en lokal utvecklingsmaskin.**

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.
2. Ger en mekanism för att separera konfiguration från kod, vilket ökar säkerheten och flexibiliteten.
3. .env-filer används ofta för att lagra miljövariabler på en lokal utvecklingsmaskin.
4. Dessa filer bör inte inkluderas i versionshanteringssystemet som Git för att skydda känslig information.

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.
2. Ger en mekanism för att separera konfiguration från kod, vilket ökar säkerheten och flexibiliteten.
3. .env-filer används ofta för att lagra miljövariabler på en lokal utvecklingsmaskin.
4. Dessa filer bör inte inkluderas i versionshanteringssystemet som Git för att skydda känslig information.
5. Används ofta för att hantera API-nycklar, databasanslutningar och andra konfigurationsdetaljer.

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.
2. Ger en mekanism för att separera konfiguration från kod, vilket ökar säkerheten och flexibiliteten.
3. .env-filer används ofta för att lagra miljövariabler på en lokal utvecklingsmaskin.
4. Dessa filer bör inte inkluderas i versionshanteringssystemet som Git för att skydda känslig information.
5. Används ofta för att hantera API-nycklar, databassolutionslutioner och andra konfigurationsdetaljer.
6. **Ger en enhetlig metod för att hantera konfiguration över olika utvecklings-, test- och produktionsmiljöer.**

Environment variables och .env-filer

1. Environment variables är nyckel-värde-par som kan användas för att konfigurera applikationer.
2. Ger en mekanism för att separera konfiguration från kod, vilket ökar säkerheten och flexibiliteten.
3. .env-filer används ofta för att lagra miljövariabler på en lokal utvecklingsmaskin.
4. Dessa filer bör inte inkluderas i versionshanteringssystemet som Git för att skydda känslig information.
5. Används ofta för att hantera API-nycklar, databassolutionslutioner och andra konfigurationsdetaljer.
6. Ger en enkelt metod för att hantera konfiguration över olika utvecklings-, test- och produktionsmiljöer.
 - Minskar risken för misstag genom att eliminera behovet av att hårdkoda värden.

Environment variables - exemple (dotenv)

```
1 import 'package:dotenv/dotenv.dart';
2
3 final env = DotEnv(includePlatformEnvironment: true)..load();
4
5 String? apiKey = env['FIREBASE_API_KEY'];
6 String? projectId = env['FIREBASE_PROJECT_ID'];
```

Environment variables - exemple (dotenv)

```
1 import 'package:dotenv/dotenv.dart';
2
3 final env = DotEnv(includePlatformEnvironment: true)..load();
4
5 String? apiKey = env['FIREBASE_API_KEY'];
6 String? projectId = env['FIREBASE_PROJECT_ID'];
```

Environment variables - exemple (dotenv)

```
1 import 'package:dotenv/dotenv.dart';
2
3 final env = DotEnv(includePlatformEnvironment: true)..load();
4
5 String? apiKey = env['FIREBASE_API_KEY'];
6 String? projectId = env['FIREBASE_PROJECT_ID'];
```

Environment variables - exemple (dotenv)

```
1 import 'package:dotenv/dotenv.dart';
2
3 final env = DotEnv(includePlatformEnvironment: true)..load();
4
5 String? apiKey = env['FIREBASE_API_KEY'];
6 String? projectId = env['FIREBASE_PROJECT_ID'];
```

```
1 // .env file in root of project (.gitignored)
2 export FIREBASE_API_KEY = "YOUR_FIREBASE_API_KEY"
3 export FIREBASE_PROJECT_ID = "YOUR_PROJECT_ID_IN_FIREBASE"
```

Nya koncept kopplade till uppgifter

Nya koncept kopplade till uppgifter

1. event-driven-architecture

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. **firebase authentication (stödjer flera login providers)**

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. firebase authentication (stödjer flera login providers)
7. environment variables

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. firebase authentication (stödjer flera login providers)
7. environment variables
8. Du får också öva på att navigera, modifiera och komplettera kod i ett existerande projekt

Nya koncept kopplade till uppgifter

1. event-driven-architecture
2. bloc
3. bloc testing
4. firebase
5. firebase firestore (realtime database)
6. firebase authentication (stödjer flera login providers)
7. environment variables
8. Du får också öva på att navigera, modifiera och komplettera kod i ett existerande projekt
 - Representativt för riktigt arbete

Dags för firebase demo!

Besök uppgifter här!

