# Build forms with GraphQL

# #whoami

**Charly POLY -** Senior Software Engineer at algolia

@whereischarly

# #whoami

I  Front-end development

# #whoami

I  Single Page Applications

# Single Page Applications

# Single Page Applications

State management

# Single Page Applications

Routing

State management

# Single Page Applications

Routing

State management

Crypto

# Single Page Applications

Routing

State management

Crypto

PWA

# Single Page Applications

Routing

State management

Crypto

Offline capabilities

PWA

# Single Page Applications

Routing

State management

Crypto

Offline capabilities

Rendering

PWA

# Single Page Applications

# Single Page Applications

Forms =

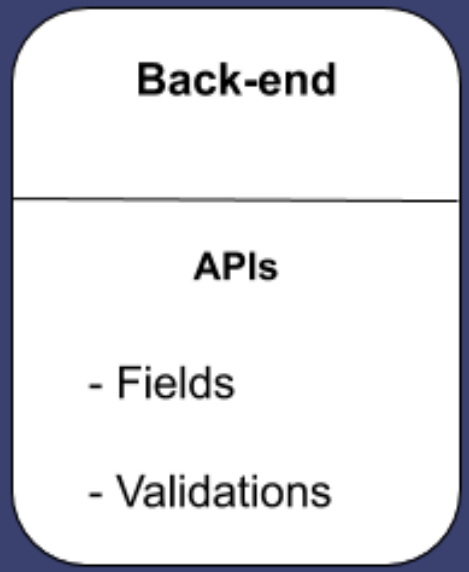
# Single Page Applications

Forms = manual and repetitive task

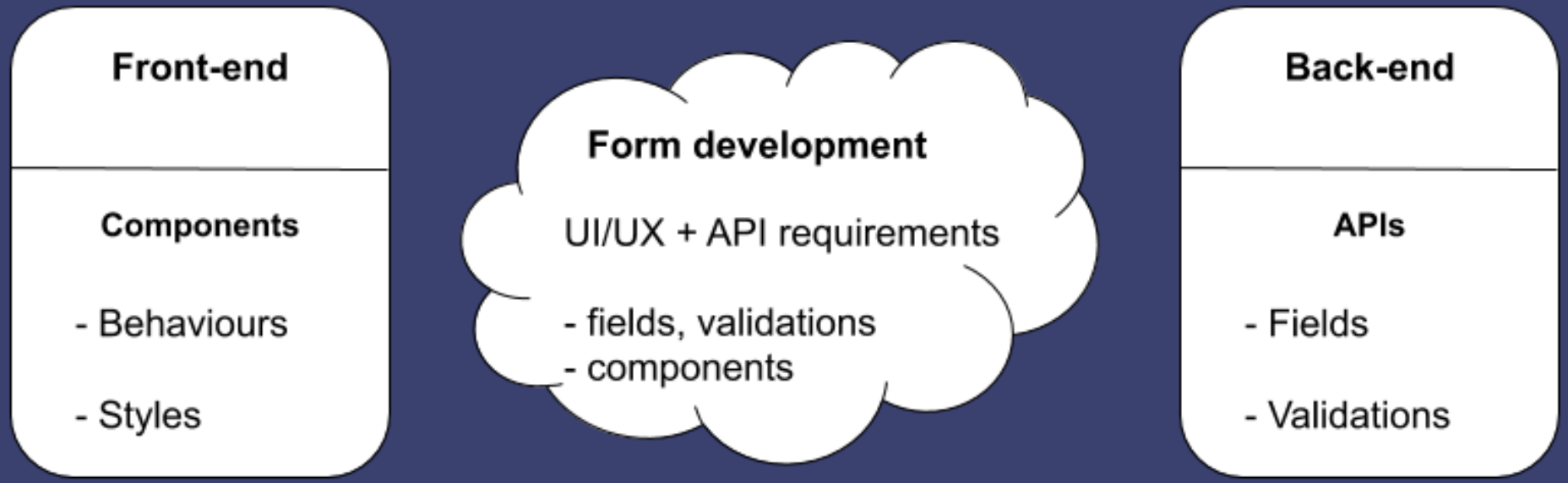
# Forms: glue between UI and APIs

# Forms: glue between UI and APIs

# Forms: glue between UI and APIs

# Forms: glue between UI and APIs

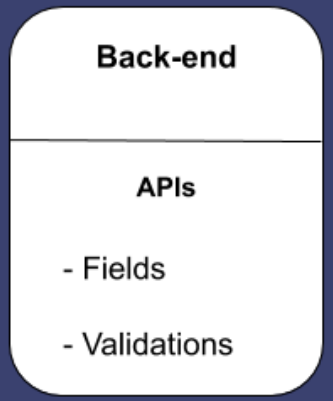


# Forms: glue between UI and APIs

```
// Render Prop
import React from 'react';
import { Formik, Form, Field, ErrorMessage } from 'formik';
import { CustomInputComponent, CustomPasswordComponent } from './common/components';
import { UserService } from './services/UserService';

const Basic = () => (
    <Formik
      validate={values => {
        let errors = {};
        if (!values.email) {
          errors.email = 'Required';
        } else if (
          !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
        ) {
          errors.email = 'Invalid email address';
        }
        return errors;
      }}
      onSubmit={(values, { setSubmitting }) => UserService.update(values, setSubmitting) }
    >
      {({ isSubmitting }) => (
        <Form>
          <Field type="email" name="email" component={CustomInputComponent} />
          <ErrorMessage name="email" component="div" />
          <Field type="password" name="password" component={CustomPasswordComponent} />
          <ErrorMessage name="password" component="div" />
          <button type="submit" disabled={isSubmitting}>
            Submit
          </button>
        </Form>
      )}
    </Formik>
);

export default Basic;
```

# Forms: glue between UI and APIs

```
1  // Render Prop
2  import React from 'react';
3  import { Formik, Form, Field, ErrorMessage } from 'formik';
4  import { CustomInputComponent, CustomPasswordComponent } from './common/components';
5  import { UserService } from './services/UserService';
6
7  const Basic = () => (
8      <Formik
9        validate={values => {
10         let errors = {};
11         if (!values.email) {
12           errors.email = 'Required';
13         } else if (
14           !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
15         ) {
16           errors.email = 'Invalid email address';
17         }
18         return errors;
19       }}
20       onSubmit={(values, { setSubmitting }) => UserService.update(values, setSubmitting) }
21     >
22       {({ isSubmitting }) => (
23         <Form>
24           <Field type="email" name="email" component={CustomInputComponent} />
25           <ErrorMessage name="email" component="div" />
26           <Field type="password" name="password" component={CustomPasswordComponent} />
27           <ErrorMessage name="password" component="div" />
28           <button type="submit" disabled={isSubmitting}>
29             Submit
30           </button>
31         </Form>
32       )}
33     </Formik>
34 );
35
36 export default Basic;
```



# Forms: glue between UI and APIs

**Front-end**

**Components**

- Behaviours
- Styles

```
1  // Render Prop
2  import React from 'react';
3  import { Formik, Form, Field, ErrorMessage } from 'formik';
4  import { CustomInputComponent, CustomPasswordComponent } from './common/components';
5  import { UserService } from './services/UserService';
6
7  const Basic = () => (
8      <Formik
9        validate={values => {
10         let errors = {};
11         if (!values.email) {
12           errors.email = 'Required';
13         } else if (
14           !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
15         ) {
16           errors.email = 'Invalid email address';
17         }
18         return errors;
19       }}
20       onSubmit={(values, { setSubmitting }) => UserService.update(values, setSubmitting) }
21     >
22       {({ isSubmitting }) => (
23         <Form>
24           <Field type="email" name="email" component={CustomInputComponent} />
25           <ErrorMessage name="email" component="div" />
26           <Field type="password" name="password" component={CustomPasswordComponent} />
27           <ErrorMessage name="password" component="div" />
28           <button type="submit" disabled={isSubmitting}>
29             Submit
30           </button>
31         </Form>
32       )}
33     </Formik>
34 );
35
36 export default Basic;
```

**Back-end**

**APIs**

- Fields
- Validations

# Forms: glue between UI and APIs

*"Forms handle the experience that users have with data"*

# Story: Improving forms development

# Speeding up form development

```
class Form extends ModuleForm {
    fields: FieldsDefinitions = {
        id: 'none',
        email: 'none',
        picture_path: {
            type: 'image', transformations: 'h_200,w_200,r_max,c_fill'
        },
        first_name: 'string*',
        last_name: 'string*',
        username: 'string*',
        job_title: 'string',
        company_name: 'string',
        language: {
            type: 'select*',
            component: LanguageSelectView,
            valueProperty: 'code',
            values: supportedLanguages,
            moduleName: 'attachment'
        }
    };

    constructor() {
        super('UserForm', 'user');
    }
}
```

# Speeding up form development

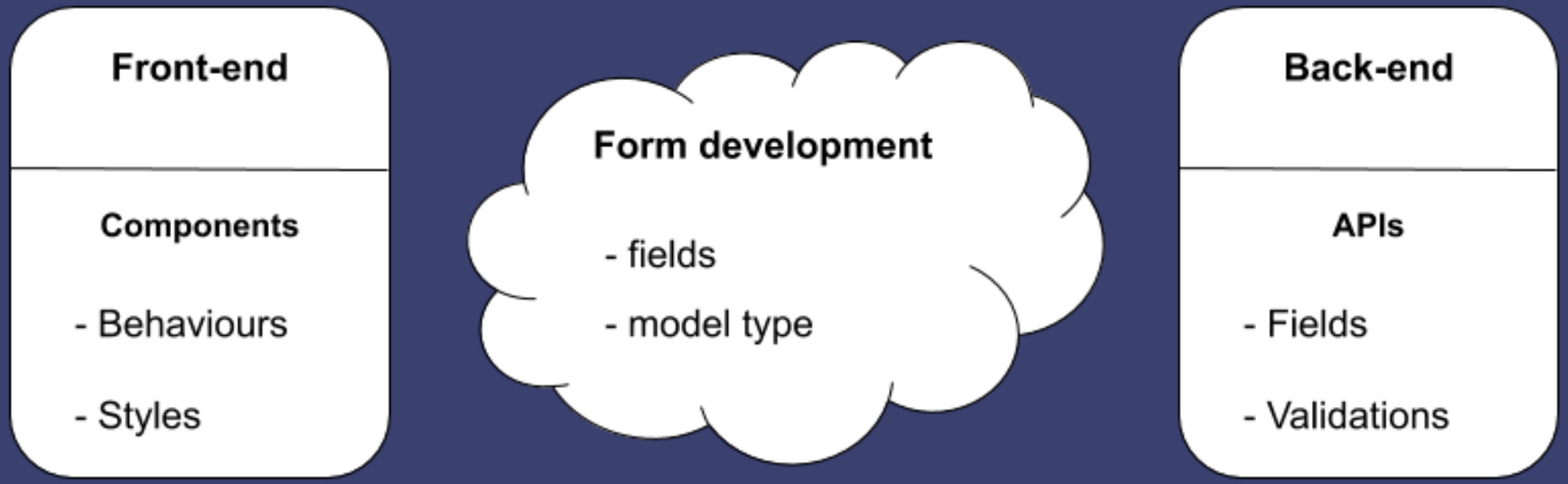
```
class Form extends ModuleForm {
    fields: FieldsDefinitions = {
        id: 'none',
        email: 'none',
        picture_path: {
            type: 'image', transformations: 'h_200,w_200,r_max,c_fill'
        },
        first_name: 'string*',
        last_name: 'string*',
        username: 'string*',
        job_title: 'string',
        company_name: 'string',
        language: {
            type: 'select*',
            component: LanguageSelectView,
            valueProperty: 'code',
            values: supportedLanguages,
            moduleName: 'attachment'
        }
    };

    constructor() {
        super('UserForm', 'user');
    }
}
```

# Speeding up form development

```
1  class Form extends ModuleForm {
2      fields: FieldsDefinitions = {
3          id: 'none',
4          email: 'none',
5          picture_path: {
6              type: 'image', transformations: 'h_200,w_200,r_max,c_fill'
7          },
8          first_name: 'string*',
9          last_name: 'string*',
10         username: 'string*',
11         job_title: 'string',
12         company_name: 'string',
13         language: {
14             type: 'select*',
15             component: LanguageSelectView,
16             valueProperty: 'code',
17             values: supportedLanguages,
18             moduleName: 'attachment'
19         }
20     };
21
22     constructor() {
23         super('UserForm', 'user');
24     }
25 }
```

# Speeding up form development

# Speeding up form development



# Speeding up form development

## Not the proper solution

# Speeding up form development

## Not the proper solution

- Too many abstractions

# Speeding up form development

## Not the proper solution

- Too many abstractions
- Not flexible

# Speeding up form development

## Not the proper solution

- Too many abstractions
- Not flexible
- Not the "React way"

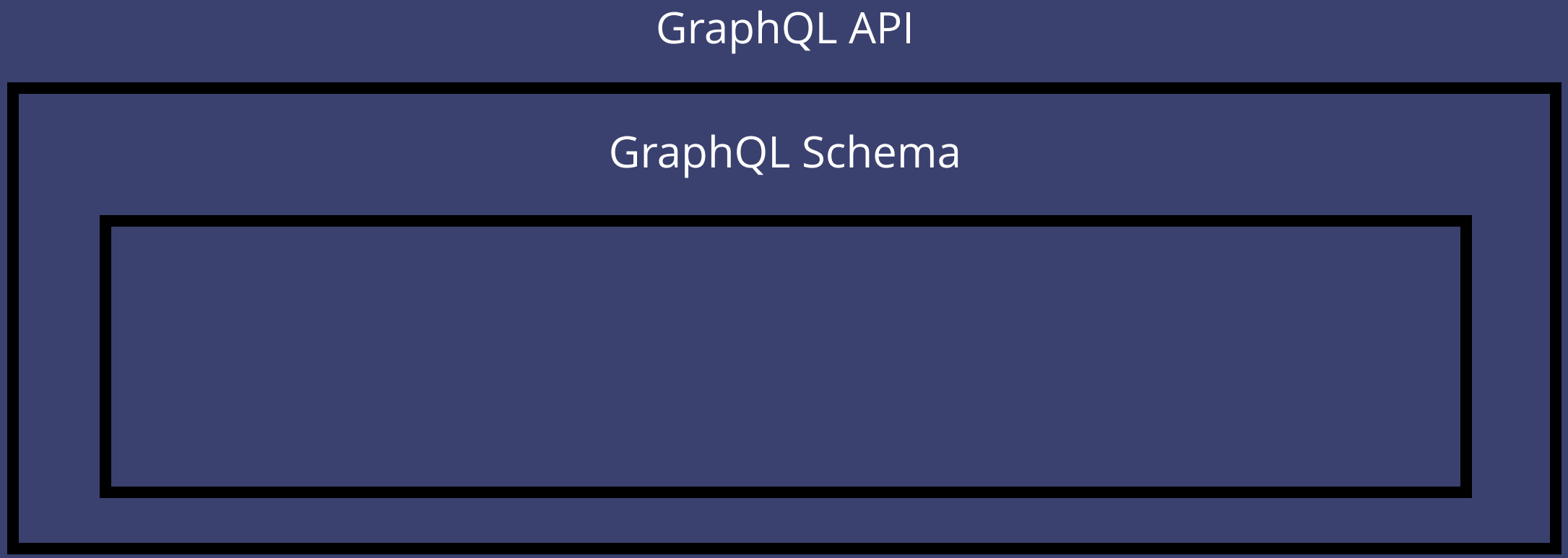
# Leveraging GraphQL to improve form development

# What is GraphQL?

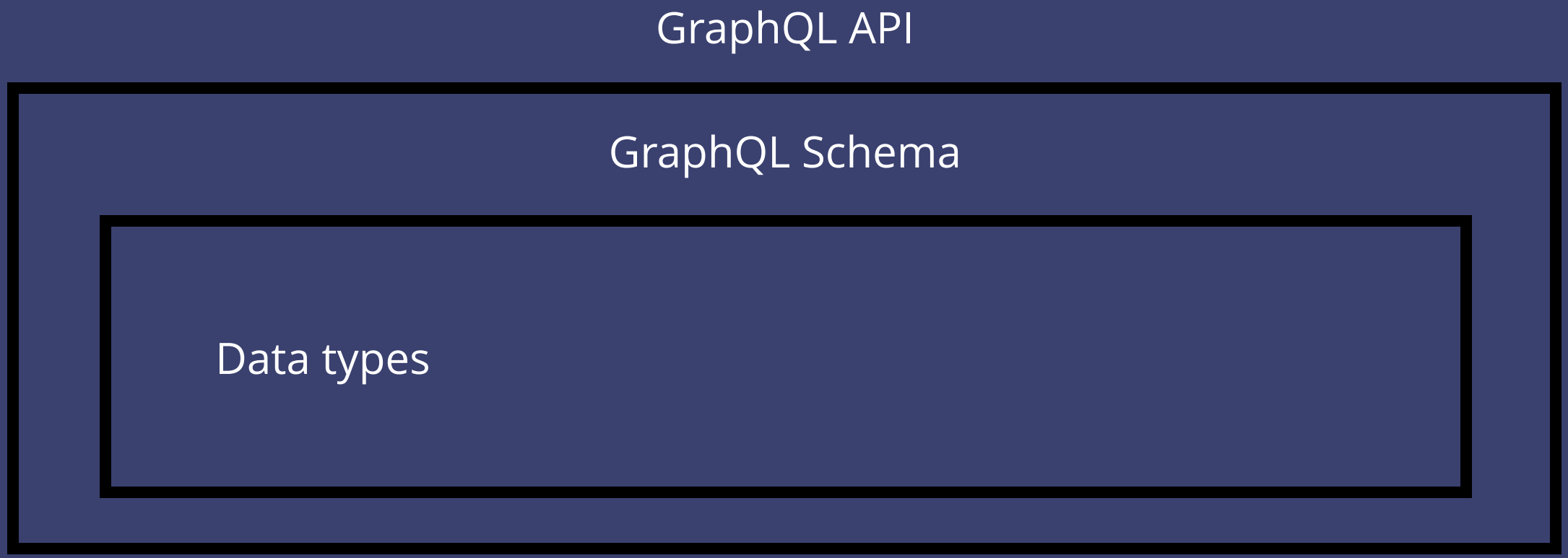
# What is GraphQL?

GraphQL API

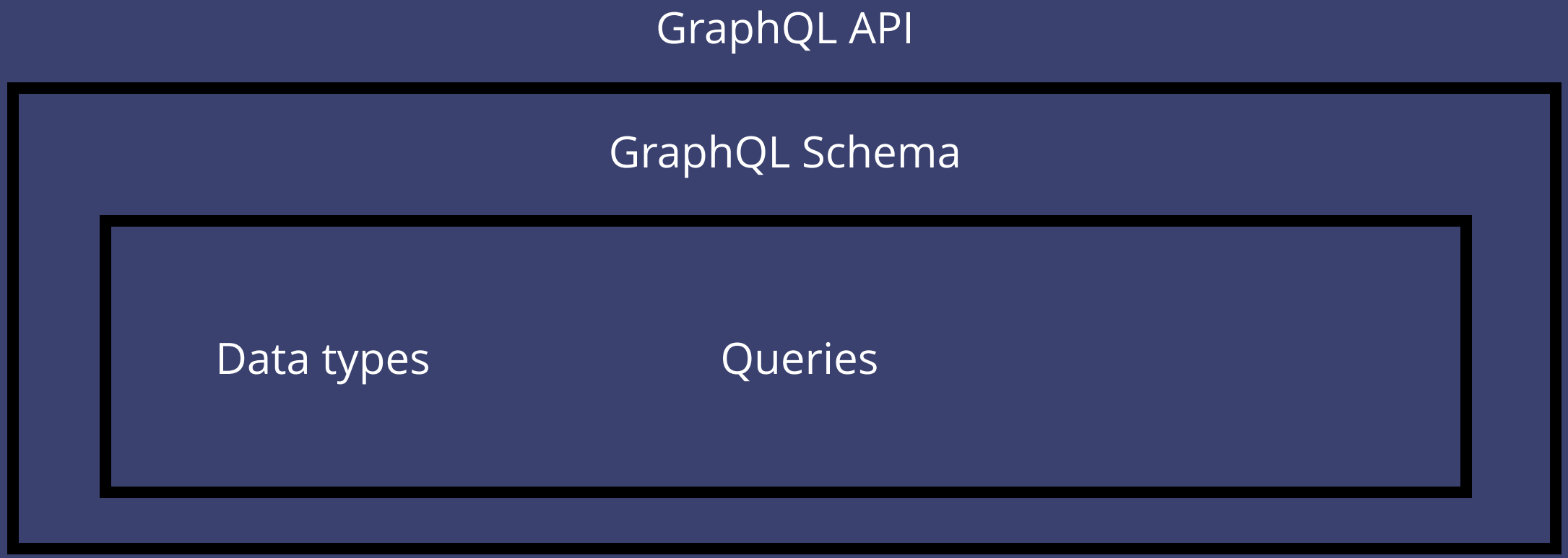
# What is GraphQL?



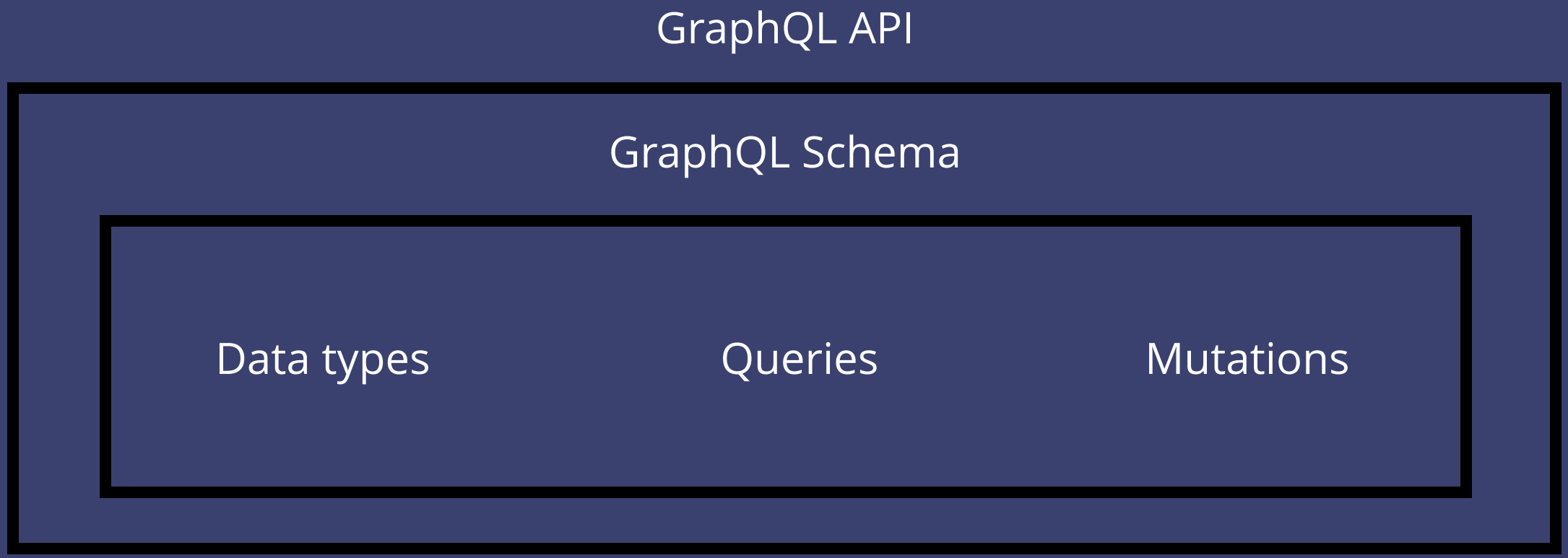
# What is GraphQL?



# What is GraphQL?



# What is GraphQL?



# How to leverage GraphQL?

## GraphQL mutation design

```
1 mutation(user: UserInputType!, company: CompanyInputType) {
2     create_account(user: $user, company: $company) {
3         user {
4             id
5         }
6     }
7 }
```

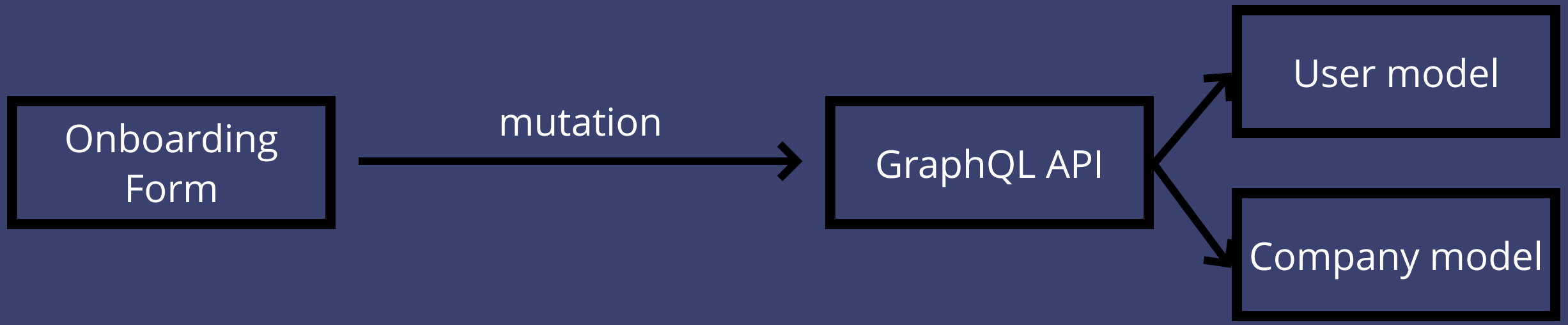
# How to leverage GraphQL?

## GraphQL mutation design

```
mutation(user: UserInputType!, company: CompanyInputType) {
    create_account(user: $user, company: $company) {
        user {
            id
        }
    }
}
```

# How to leverage GraphQL?

## 1. GraphQL mutations are based on business logic



# How to leverage GraphQL?

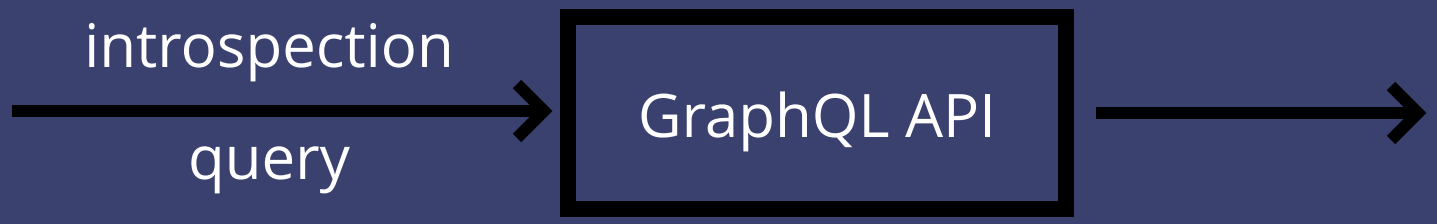
## 2. GraphQL introspection

# How to leverage GraphQL?

## 2. GraphQL introspection

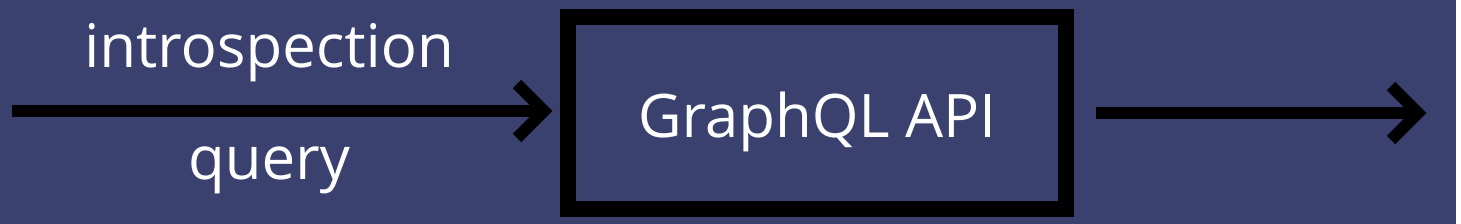
# How to leverage GraphQL?

## 2. GraphQL introspection



# How to leverage GraphQL?

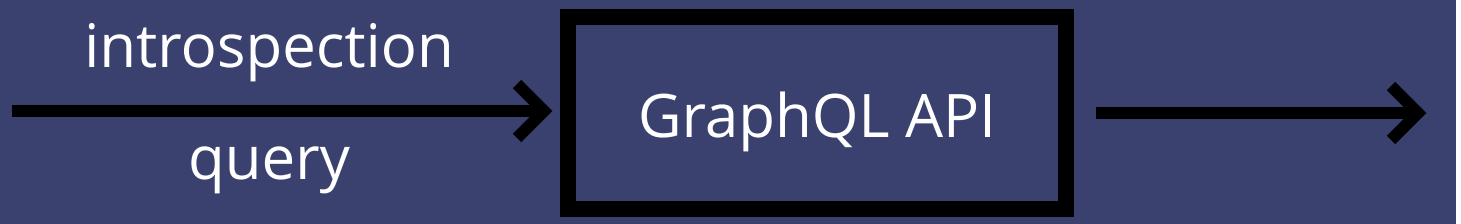
## 2. GraphQL introspection




```
{
  "name": "createAccount",
  "__typename": "__Field",
  "isDeprecated": false,
  "deprecationReason": null,
  "args": [
    {
      "name": "user",
      "type": {
        "kind": "NON_NULL",
        "name": null,
        "ofType": {
          "kind": "InputType",
          "name": "UserInputType",
          "ofType": null,
          "__typename": "__Type"
        },
        "__typename": "__Type"
      },
      "defaultValue": null,
      "__typename": "__InputValue"
    },
  // ...
}
```

# How to leverage GraphQL?

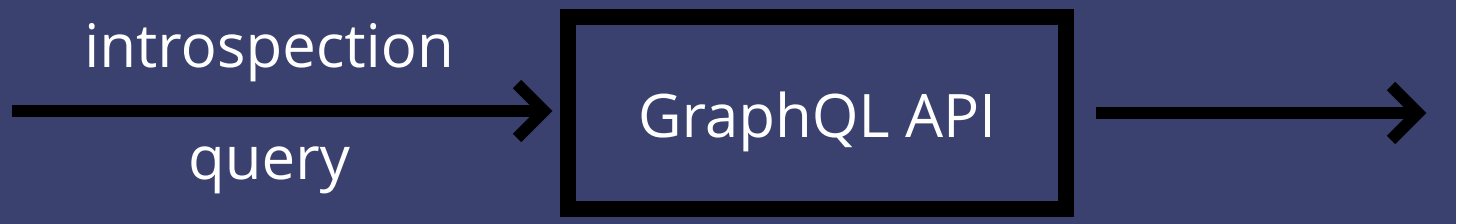
## 2. GraphQL introspection




```
{
  "name": "createAccount",
  "__typename": "__Field",
  "isDeprecated": false,
  "deprecationReason": null,
  "args": [
    {
      "name": "user",
      "type": {
        "kind": "NON_NULL",
        "name": null,
         "ofType": {
            "kind": "InputType",
            "name": "UserInputType",
            "ofType": null,
            "__typename": "__Type"
          },
        "__typename": "__Type"
      },
      "defaultValue": null,
      "__typename": "__InputValue"
    },
  // ...
}
```

# How to leverage GraphQL?

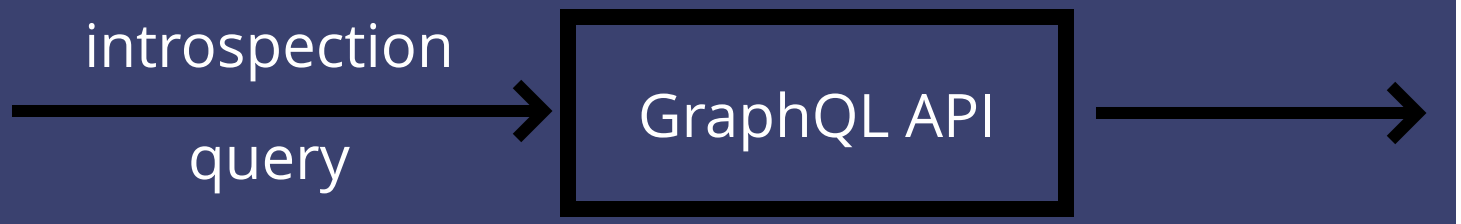
## 2. GraphQL introspection




```
{
  "name": "createAccount",
  "__typename": "__Field",
  "isDeprecated": false,
  "deprecationReason": null,
  "args": [
    {
      "name": "user",
      "type": {
        "kind": "NON_NULL",
        "name": null,
        "ofType": {
          "kind": "InputType",
          "name": "UserInputType",
          "ofType": null,
          "__typename": "__Type"
        },
        "__typename": "__Type"
      },
      "defaultValue": null,
      "__typename": "__InputValue"
    },
  // ...
}
```

# How to leverage GraphQL?

## 2. GraphQL introspection




```
{
  "name": "createAccount",
  "__typename": "__Field",
  "isDeprecated": false,
  "deprecationReason": null,
  "args": [
    {
      "name": "user",
      "type": {
        "kind": "NON_NULL",
        "name": null,
        "ofType": {
          "kind": "InputType",
          "name": "UserInputType",
          "ofType": null,
          "__typename": "__Type"
        },
        "__typename": "__Type"
      },
      "defaultValue": null,
      "__typename": "__InputValue"
    },
  // ...
}
```

# How to leverage GraphQL?

## 2. GraphQL introspection

# GraphQL for forms

# GraphQL for forms

Mutation =

# GraphQL for forms

Mutation = similar to Form UI

# GraphQL for forms

Mutation = similar to Form UI

Mutation name =

# GraphQL for forms

Mutation = similar to Form UI

Mutation name = fields + requirements

# GraphQL for forms

# <Frontier />

@wittydeveloper @chmelevskij

# <Frontier />

## Simplicity

# Frontier forms

## Simplicity

```
<Frontier mutation={mutation} client={client} initialValues={{ user: { email: 'hello@charlypoly.com ' } }}>
  {
    ({ state, modifiers, form }) => {
      return (
        <form onSubmit={modifiers.save}>
          <h2>Create a user</h2>
          <p>
            <label htmlFor="name">Name*</label> <br />
            <input
              type="text"
              name="name"
              value={state.values.user.name} onChange={modifiers.user.name.change}
            />
            {
              state.errors.user && state.errors.user.name &&
              <p>
                Error: "{state.errors.user.name}"
              </p>
            }
          </p>
          <p>
            <input type="submit" value="Save" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

Simplicity

```
<Frontier mutation={mutation} client={client} initialValues={{ user: { email: 'hello@charlypoly.com ' } }}>
  {
    ({ state, modifiers, form }) => {
      return (
        <form onSubmit={modifiers.save}>
          <h2>Create a user</h2>
          <p>
            <label htmlFor="name">Name*</label> <br />
            <input
              type="text"
              name="name"
              value={state.values.user.name} onChange={modifiers.user.name.change}
            />
            {
              state.errors.user && state.errors.user.name &&
              <p>
                Error: "{state.errors.user.name}"
              </p>
            }
          </p>
          <p>
            <input type="submit" value="Save" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

## Simplicity

```
<Frontier mutation={mutation} client={client} initialValues={{ user: { email: 'hello@charlypoly.com ' } }}>
  {
    ({ state, modifiers, form }) => {
      return (
        <form onSubmit={modifiers.save}>
          <h2>Create a user</h2>
          <p>
            <label htmlFor="name">Name*</label> <br />
            <input
              type="text"
              name="name"
              value={state.values.user.name} onChange={modifiers.user.name.change}
            />
            {
              state.errors.user && state.errors.user.name &&
              <p>
                Error: "{state.errors.user.name}"
              </p>
            }
          </p>
          <p>
            <input type="submit" value="Save" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

## Simplicity

```
<Frontier mutation={mutation} client={client} initialValues={{ user: { email: 'hello@charlypoly.com ' } }}>
  {
    ({ state, modifiers, form }) => {
      return (
        <form onSubmit={modifiers.save}>
          <h2>Create a user</h2>
          <p>
            <label htmlFor="name">Name*</label> <br />
            <input
              type="text"
              name="name"
              value={state.values.user.name} onChange={modifiers.user.name.change}
            />
            {
              state.errors.user && state.errors.user.name &&
              <p>
                Error: "{state.errors.user.name}"
              </p>
            }
          </p>
          <p>
            <input type="submit" value="Save" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

## Simplicity

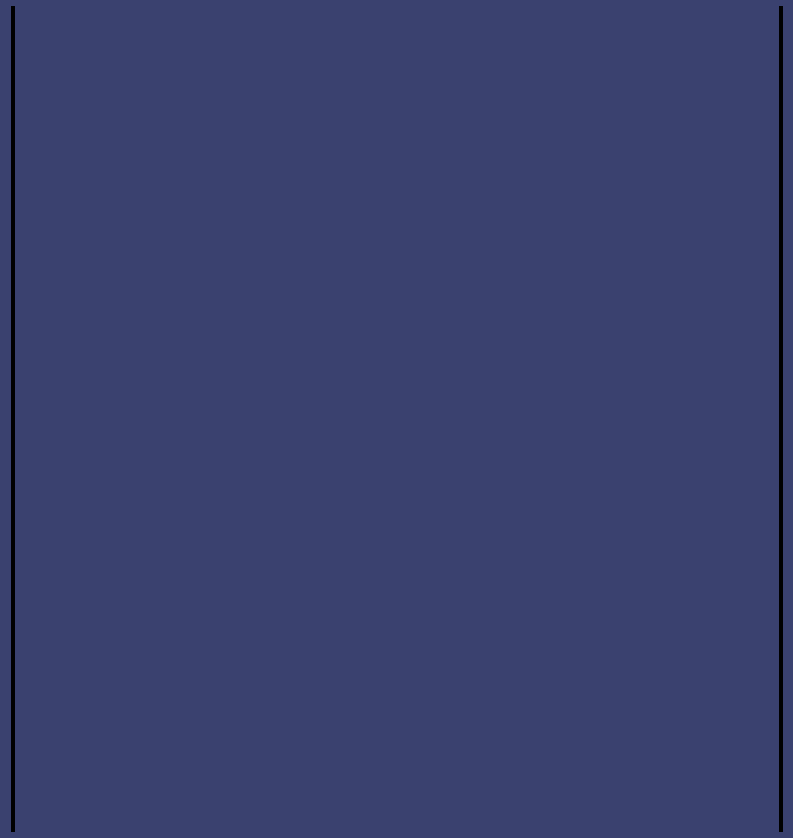
```
<Frontier mutation={mutation} client={client} initialValues={{ user: { email: 'hello@charlypoly.com ' } }}>
  {
    ({ state, modifiers, form }) => {
      return (
        <form onSubmit={modifiers.save}>
          <h2>Create a user</h2>
          <p>
            <label htmlFor="name">Name*</label> <br />
            <input
              type="text"
              name="name"
              value={state.values.user.name} onChange={modifiers.user.name.change}
            />
            {
              state.errors.user && state.errors.user.name &&
              <p>
                Error: "{state.errors.user.name}"
              </p>
            }
          </p>
          <p>
            <input type="submit" value="Save" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# <Frontier />

## Full data lifecycle management

# Frontier forms

## Full-data lifecycle management



# Frontier forms

## Full-data lifecycle management

Fields definitions

# Frontier forms

## Full-data lifecycle management

Fields definitions

Form state

# Frontier forms

## Full-data lifecycle management

Fields definitions | Form state | Save data

# Frontier forms

## Full-data lifecycle management

Fields definitions

GraphQL

Form state

Save data

# Frontier forms

## Full-data lifecycle management

Fields definitions

GraphQL

Form state

final-form

Save data

# Frontier forms

## Full-data lifecycle management

| Fields definitions | Form state | Save data |
| --- | --- | --- |
| GraphQL | final-form | Apollo GraphQL |

# <Frontier />

## UI-Kit

# Frontier forms

UI-Kit

```
const mutation = gql`
  mutation ($user: UserInputType!) {
    createUser(user: $user) {
      ...User
    }
  }
`;

<Frontier mutation={mutation} client={client} uiKit={ApplicationUIkit} />
```

# Frontier forms

UI-Kit

```
const mutation = gql`
  mutation ($user: UserInputType!) {
    createUser(user: $user) {
      ...User
    }
  }
`;

<Frontier mutation={mutation} client={client} uiKit={ApplicationUIkit} />
```

# Frontier forms

UI-Kit

```
1  const mutation = gql`
2    mutation ($user: UserInputType!) {
3      createUser(user: $user) {
4        ...User
5      }
6    }
7  `;
8
9  <Frontier mutation={mutation} client={client} uiKit={ApplicationUIkit} />
```

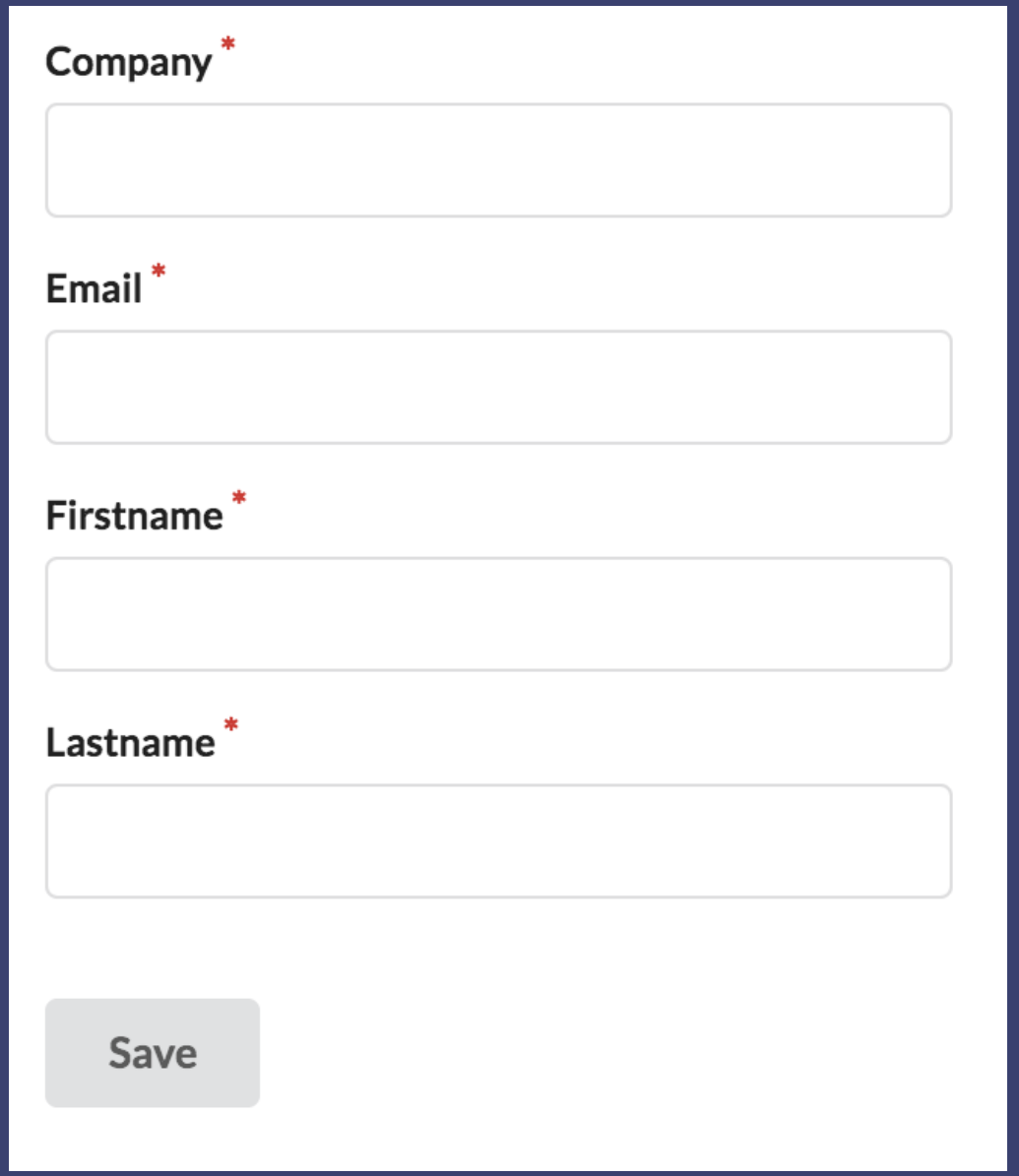
# Frontier forms

UI-Kit

```
const mutation = gql`
  mutation ($user: UserInputType!) {
    createUser(user: $user) {
      ...User
    }
  }
`;

<Frontier mutation={mutation} client={client} uiKit={ApplicationUIkit} />
```

# Frontier forms

UI-Kit



# <Frontier />

## Flexible

# Frontier forms

## Flexible

```
<Frontier client={client} mutation={mutation} uiKit={ApplicationUIkit}>
  {
    ({ form, kit }) => {
      return (
        <form className='ui form' onSubmit={(e) => { e.preventDefault(); form.submit(); }}>
          <div>
            {kit.company()}
          </div>
          <Message
            info
            header='Is my company already registered?'
            list={[
              'If your company is already registred under a Business plan, please do register using the Business form',
            ]}
          />
          <br />
          <br />
          <div>
            {kit.email()}
          </div>
          <br />
          <div>
            {kit.firstname()}
          </div>
          <br />
          <div>
            {kit.lastname()}
          </div>
          <p>
            <input type="submit" value="Save" className="ui button" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

## Flexible

```
<Frontier client={client} mutation={mutation} uiKit={ApplicationUIkit}>
{
  ({ form, kit }) => {
    return (
      <form className='ui form' onSubmit={(e) => { e.preventDefault(); form.submit(); }}>
        <div>
          {kit.company()}
        </div>
        <Message
          info
          header='Is my company already registered?'
          list={[
            'If your company is already registred under a Business plan, please do register using the Business form',
          ]}
        />
        <br />
        <br />
        <div>
          {kit.email()}
        </div>
        <br />
        <div>
          {kit.firstname()}
        </div>
        <br />
        <div>
          {kit.lastname()}
        </div>
        <p>
          <input type="submit" value="Save" className="ui button" />
        </p>
      </form>
    )
  }
}
</Frontier>
```

# Frontier forms

## Flexible

```
<Frontier client={client} mutation={mutation} uiKit={ApplicationUIkit}>
  {
    ({ form, kit }) => {
      return (
        <form className='ui form' onSubmit={(e) => { e.preventDefault(); form.submit(); }}>
          <div>
            {kit.company()}
          </div>
          <Message
            info
            header='Is my company already registered?'
            list={[
              'If your company is already registred under a Business plan, please do register using the Business form',
            ]}
          />
          <br />
          <br />
          <div>
            {kit.email()}
          </div>
          <br />
          <div>
            {kit.firstname()}
          </div>
          <br />
          <div>
            {kit.lastname()}
          </div>
          <p>
            <input type="submit" value="Save" className="ui button" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

## Flexible

```
<Frontier client={client} mutation={mutation} uiKit={ApplicationUIkit}>
  {
    ({ form, kit }) => {
      return (
        <form className='ui form' onSubmit={(e) => { e.preventDefault(); form.submit(); }}>
          <div>
            {kit.company()}
          </div>
          <Message
            info
            header='Is my company already registered?'
            list={[
              'If your company is already registred under a Business plan, please do register using the Business form',
            ]}
          />
          <br />
          <br />
          <div>
            {kit.email()}
          </div>
          <br />
          <div>
            {kit.firstname()}
          </div>
          <br />
          <div>
            {kit.lastname()}
          </div>
          <p>
            <input type="submit" value="Save" className="ui button" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

## Flexible

```
<Frontier client={client} mutation={mutation} uiKit={ApplicationUIkit}>
  {
    ({ form, kit }) => {
      return (
        <form className='ui form' onSubmit={(e) => { e.preventDefault(); form.submit(); }}>
          <div>
            {kit.company()}
          </div>
          <Message
            info
            header='Is my company already registered?'
            list={[
              'If your company is already registred under a Business plan, please do register using the Business form',
            ]}
          />
          <br />
          <br />
          <div>
            {kit.email()}
          </div>
          <br />
          <div>
            {kit.firstname()}
          </div>
          <br />
          <div>
            {kit.lastname()}
          </div>
          <p>
            <input type="submit" value="Save" className="ui button" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

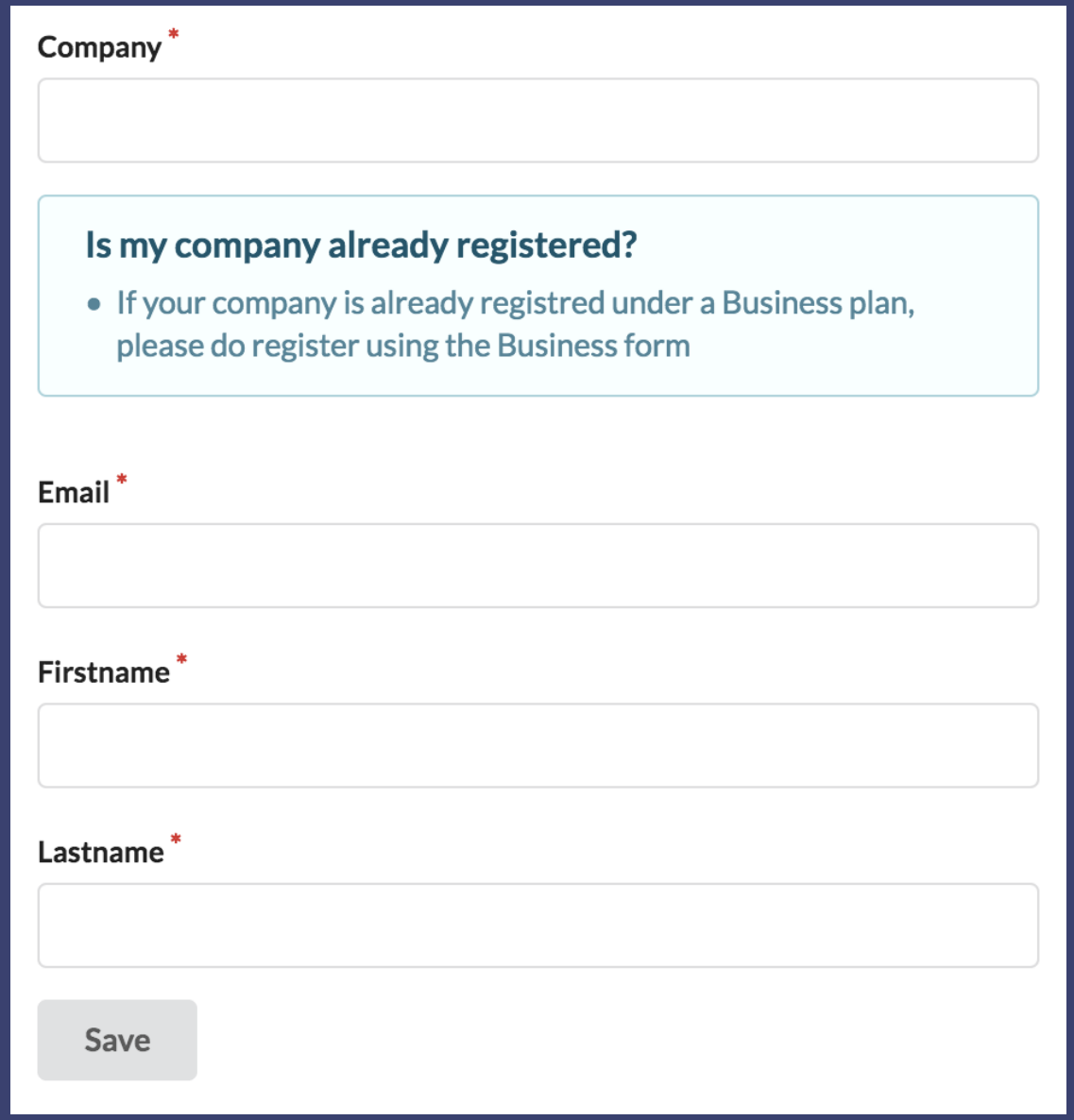
# Frontier forms

## Flexible

```
<Frontier client={client} mutation={mutation} uiKit={ApplicationUIkit}>
  {
    ({ form, kit }) => {
      return (
        <form className='ui form' onSubmit={(e) => { e.preventDefault(); form.submit(); }}>
          <div>
            {kit.company()}
          </div>
          <Message
            info
            header='Is my company already registered?'
            list={[
              'If your company is already registred under a Business plan, please do register using the Business form',
            ]}
          />
          <br />
          <br />
          <div>
            {kit.email()}
          </div>
          <br />
          <div>
            {kit.firstname()}
          </div>
          <br />
          <div>
            {kit.lastname()}
          </div>
          <p>
            <input type="submit" value="Save" className="ui button" />
          </p>
        </form>
      )
    }
  }
</Frontier>
```

# Frontier forms

Flexible



# Frontier forms

## A new form development experience

# Frontier forms

## A new form development experience

- **Full data lifecycle management**: state, validation, save

# Frontier forms

## A new form development experience

- **Full data lifecycle management**: state, validation, save
- **UI-kit full rendering:** bring a consistent experience to your users

# Frontier forms

## A new form development experience

- **Full data lifecycle management**: state, validation, save
- **UI-kit full rendering:** bring a consistent experience to your users
- **Simple and iterative**: choose your form development flow

# Frontier forms: demos

**How to:**

- Define and use a UI-Kit for your application
- Advanced use-cases

  (validations & complex types)

https://codesandbox.io/s/7j3xo3qy26

# Frontier: built for the future

# Frontier: built for the future

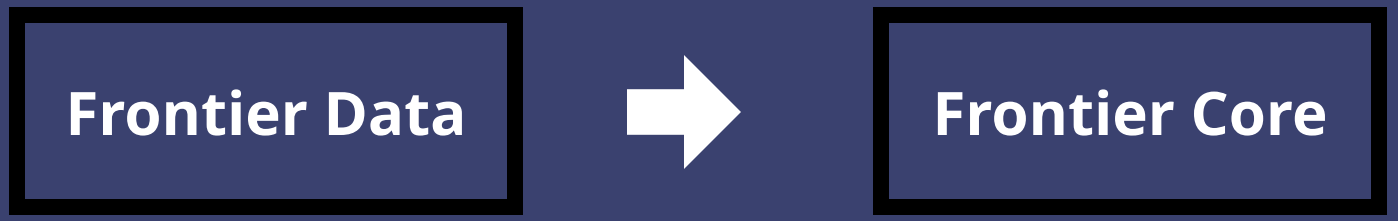
**Frontier Data**

# Frontier: built for the future

**Frontier Data**

- Apollo GraphQL

# Frontier: built for the future




- Apollo GraphQL

# Frontier: built for the future

**Frontier Data**

**Frontier Core**

- Apollo GraphQL

- ajv
- final-form

# Frontier: built for the future


- Apollo GraphQL

- ajv
- final-form

# Frontier: built for the future

# Frontier: built for the future

- Swagger support

# Frontier: built for the future

- Swagger support
- More flavours: Vue.js, Angular and React Native

# Frontier: built for the future

- Swagger support
- More flavours: Vue.js, Angular and React Native
- API improvements

# frontier-forms.dev 👀

# Frontier forms

Data-driven forms that let you focus on what matters: your application.

Provide a `GraphQL` mutation and `<Frontier/>` will do the rest for you.

Both fast to use and performant ⚡ !

```
import gql from "graphql-tag";
import { Frontier } from "frontier-forms";
import { myApplicationKit } from "./uiKit";
import { client } from "./apollo-client";

const mutation = gql`
    mutation($user: User!) {
      createUser(user: $user) { id }
    }
`;
```

We are hiring!

honest.engineering

@whereischarly

/wittydeveloper

# Thank you!

We are hiring!

honest.engineering

@whereischarly

/wittydeveloper