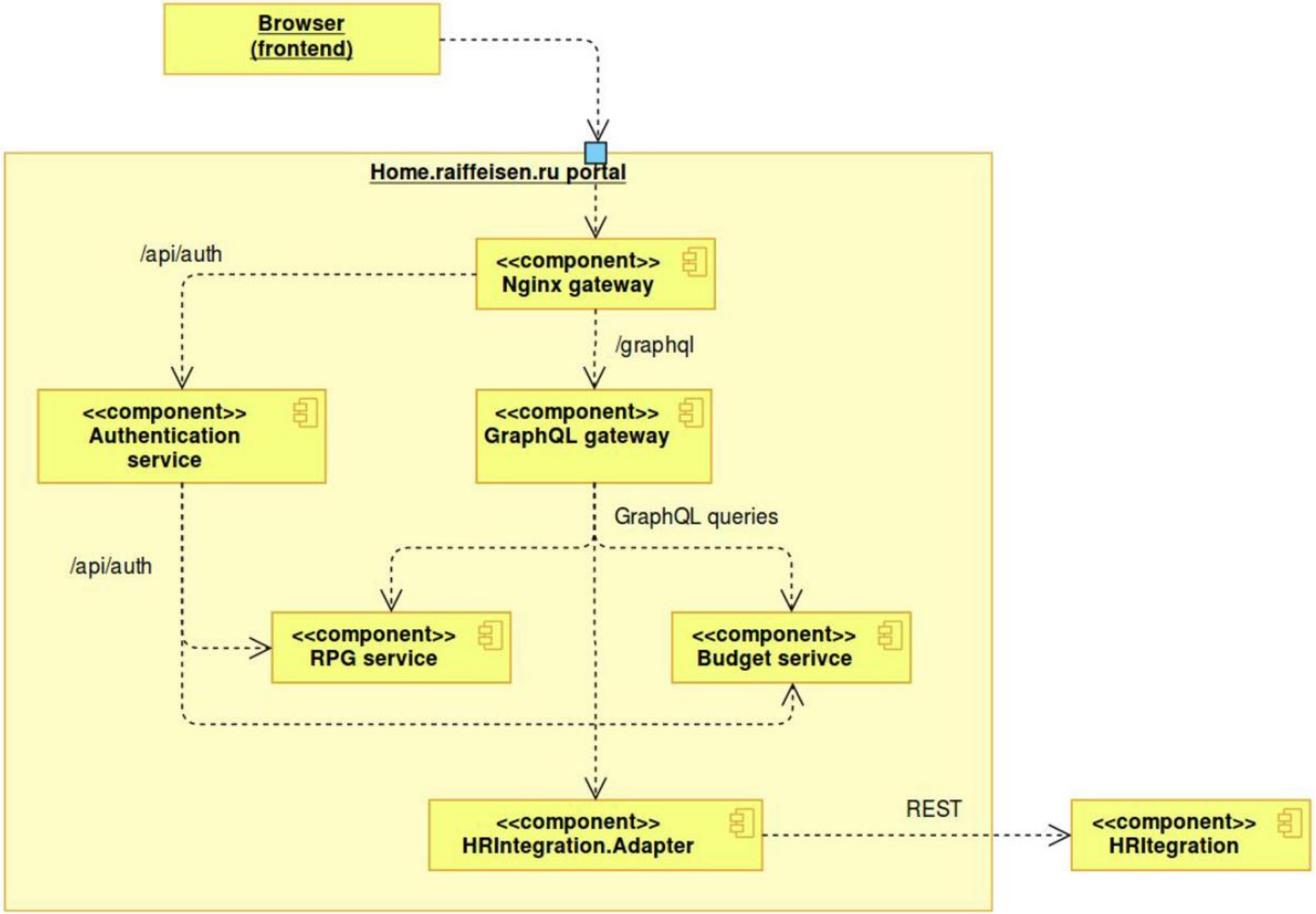
Райффайзен БАНК

Backend notes about GraphQL

```
query {
    success {
        story
    }
}
```

OGORODNIKOV Sergey
Senior .Net developer
CnB Team

Просто. Четко.
Компетентно.


Browser
(frontend)
Home.raiffeisen.ru portal
/api/auth
<<component>>
Nginx gateway
/graphql
<<component>>
Authentication
service
<<component>>
GraphQL gateway
GraphQL queries
/api/auth
<<component>>
RPG service
<<component>>
Budget serivce
<<component>>
HRIntegration.Adapter
REST
<<component>>
HRItegration

Budget service GraphQL schema

Search the docs ...

budgetGetDepartment(...): BudgetDepartmentMembers

budgetLinearCostCentersForDepartmer ...): [BudgetLinearCostDtoCenter]

budgetAvailableLinearCostCentersForD ...): [BudgetLinearCostDtoCenter]

budgetAvailableLinearCostCentersSubA [BudgetLinearCostCenterSubAreas]

budgetLinearCostCenterStatistics(...): [budgetGetLinearCostCentersStatisticsR

budgetFrontendConfiguration: budgetFrontendConfigurationSection

budgetGetTeamCostCentersStatistics(...): budgetGetTeamCostCentersStatisticsQu

budgetGetTotalStatistics(...): budgetStatisticsResult

budgetExportCostCenterStatistics(...): GuidGraphType

budgetExportEmployeeStatistics(...): GuidGraphType

budgetSearchEmployees(...): [budgetEmployeeSlimType]

budgetGetStatisticFilterAvailableValues: budgetGetStatisticFilterAvailableValues(

getComments(...): [Comment]

MUTATIONS

budgetGetDepartment(
departmentId: Int
): BudgetDepartmentMembers

TYPE DETAILS

```
type BudgetDepartmentMembers {
  id: Int!
  name: String!
  canAdd: Boolean!
  employees: [BudgetEmployee]
  fteList: [BudgetFte]
  ahcTotalStatistic: [BudgetCostCenterSlimStatistic]
}
```

ARGUMENTS

departmentId: Int

TYPE DETAILS

```
type BudgetEmployee {
  id: Int!
  employeeId: Int!
  employeeNumber: String!
  fullName: String!
  login: String
  startDate: Date
  endDate: Date
  isOnMaternityLeave: Boolean!
  isPrimaryJob: Boolean!
  canEdit: Boolean!
  justification: String
  department: budgetDepartmentSlim
  position: String!
  teamsEngagement: String!
  role: BudgetRole
  profile: BudgetProfile
  linearCostCenter: BudgetLinearCostDtoCenter
  teamCostCenter: BudgetTeamCostDtoCenter
  createdBy: budgetEmployeeSlimType
  editedBy: budgetEmployeeSlimType
}
```

linearCostCenter: BudgetLinearCostDtoCenter

TYPE DETAILS

```
type BudgetLinearCostDtoCenter {
  id: Int!
  code: String!
  name: String!
  town: String
  townEng: String
  teamCostCenters: [BudgetTeamCostDtoCenter]
}
```

teamCostCenters: [BudgetTeamCostDtoCenter]

TYPE DETAILS

```
type BudgetTeamCostDtoCenter {
  id: Int!
  code: String!
  name: String!
  town: String
  townEng: String
  teamId: Int!
}
```

Classic backend 3-layer architecture

Infrastructure

maps business objects to DTO to be serialized to JSON

↓

Logic

converts them to business objects

↓

Data

gets Entities from DB

GetTeam

```
class DepartmentDto
{
    int Id;
    string Name;
    EmployeeDto Manager;
    DepartmentDto ParentDepartment;
    EmployeeDto[] Employees;
}
```

```
class EmployeeDto
{
    int Id;
    string Fio;
    DepartmentDto Department;
    EmployeeDto Manager;
}
```

```
class TeamDto
{
    int Id;
    string Name;
    EmployeeDto[] Employees;
}
```

GetTeams, GetAvailableTeams

```
class DepartmentDto
{
    int Id;
    string Name;
    EmployeeDto Manager;
    DepartmentDto ParentDepartment;
    EmployeeDto[] Employees;
}
```

```
class EmployeeDto
{
    int Id;
    string Fio;
    DepartmentDto Department;
    EmployeeDto Manager;
}
```

```
class TeamDto
{
    int Id;
    string Name;
    EmployeeDto[] Employees;
}
```

```
class DepartmentSlimDto
{
    int Id;
    string Name;
}
```

```
class EmployeeSlimDto
{
    int Id;
    string Fio;
    DepartmentDtoSlim Department;
}
```

```
class TeamSlimDto
{
    int Id;
    string Name;
    EmployeeSlimDto[] Employees;
}
```

```
class TeamSuperSlimDto
{
    int Id;
    string Name;
}
```

GraphQL + CQRS

GraphQL Query

```
public sealed class TeamSlimType : ObjectGraphType<Team>
{
    public TeamSlimType()
    {
        Name = "budgetTeamSlimType";
        Description = "Slim information about team";
        Field(t => t.Id).Description("Team identifier in budget system");
        Field(t => t.Name).Description("Team name");
    }
}
```

Application Command

```
public class GetAvailableTeamsQueryResult
{
    public List<Team> AvailableTeams { get; set; }

    public bool CanAdd { get; set; }
}
```

Domain Entities

```
public class Team : IEntityKey<int>
{
    public virtual int Id { get; set; }

    public virtual string Name { get; set; }

    public virtual string BusinessDomain { get; set; }

    public virtual TeamType TeamType { get; set; }

    public virtual ISet<EmployeeToTeam> Employees { get; set; }

    public virtual ISet<TeamManager> TeamManagers { get; set; }
}
```

Full team type

```
public sealed class TeamFullType : ObjectGraphType<Team>
{
    public TeamFullType()
    {
        Name = "budgetTeamSlimType";
        Description = "Slim information about team";
        Field(t => t.Id).Description("Team identifier in budget system");
        Field(t => t.Name).Description("Team name");
        Field<EnumerationGraphType<ApprovalStatusEnum>>(t => t.ApprovalStatus)
            .Description("Approval status");
        Field<ListGraphType<EmployeeToTeamType>>(t => t.Employees)
            .Description("Team members collection");
        Field<ListGraphType<TeamManagerType>>(t => t.TeamManagers)
            .Description("Team managers collection");
    }
}
```


WE NEED TO GO
DEEPER
memegenerator.net

N+1 problem

```
{ teams { id; name; } }
```

Select * from teams

```
{
    teams {
        id
        name
        employees { id name }
    }
}
```

Select * from teams

Select * from employees where teamId = 1

Select * from employees where teamId = 2

…..

Select * from employees where teamId = N

N+1 problem salvation

- **Prefetch**

```
var department = await _entityFactoryService.Create<Department>()
    .Query()
    .Where(d => d.Id == message.DepartmentId)
    .FetchMany(d => d.Employees).ThenFetch(ed => ed.Employee)
    .FetchMany(d => d.Ftes).ThenFetch(f => f.LinearCostCenter).ThenFetchMany(c => c.TeamCostCenters)
    .SingleOrDefaultAsync(cancellationToken);
```

- - **IDataLoader**

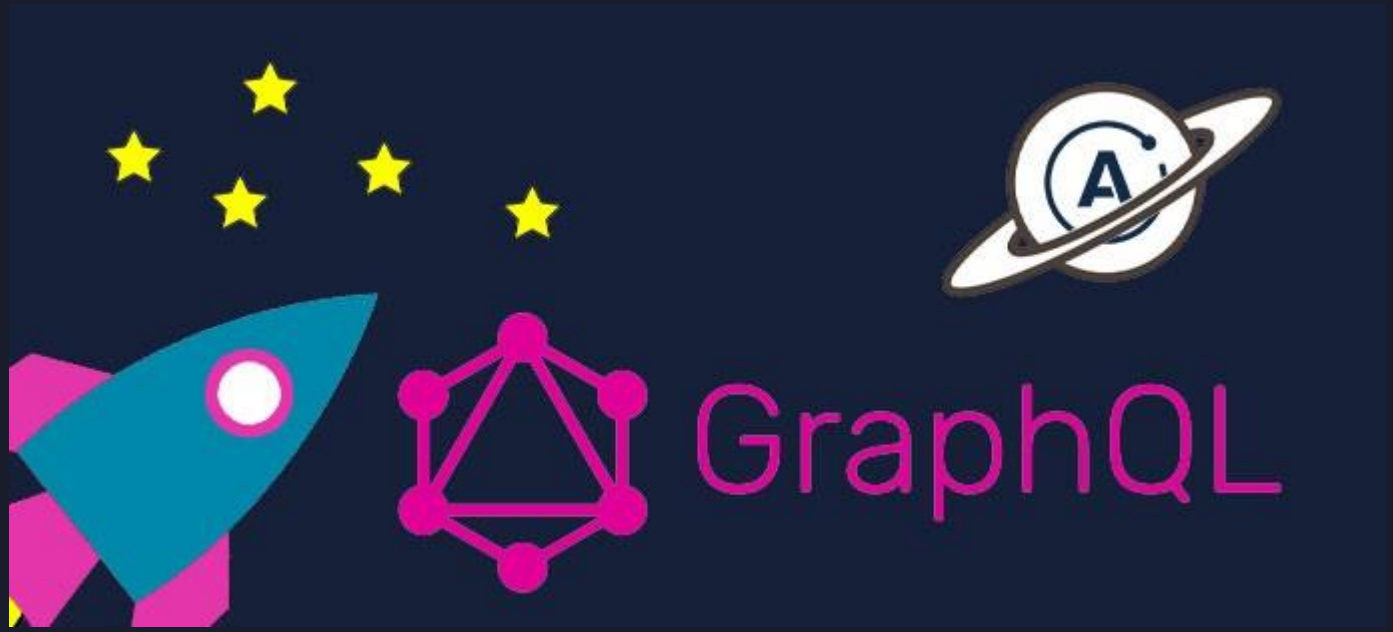
  Interface IDataLoader<T> { Task<ICollection<T>> LoadAsync(params TKey[] keys); }

Resume

- Pros:
  - Traffic reduction++
- Cons:
  - Too new? Not usual?

Райффайзен БАНК

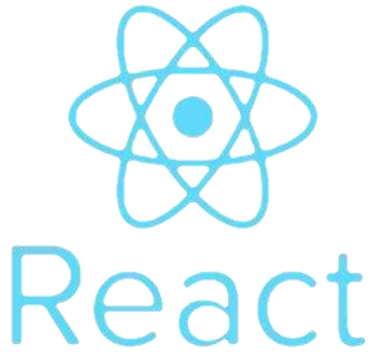
Frontend notes about GraphQL

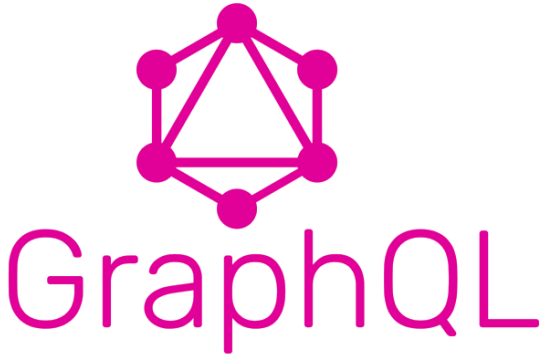


LATYPOVA Alia
Senior .Net developer
CnB Team

Просто. Четко.
Компетентно.

Recipe





Apollo Client

Apollo Client is a complete state management library for JavaScript apps.

Features

•**Declarative data fetching:** Write a query and receive data without manually tracking loading states

•**Excellent developer experience:** Enjoy helpful tooling for TypeScript, Chrome DevTools, and VS Code

•**Designed for modern React:** Take advantage of the latest React features, such as hooks

•**Incrementally adoptable:** Drop Apollo into any JavaScript app seamlessly

•**Universally compatible:** Use any build setup and any GraphQL API

•**Community driven:** Share knowledge with thousands of developers, thanks to our active open source community

Get started

```
npm install apollo-boost @apollo/react-hooks graphql

import ApolloClient from 'apollo-boost';

const client = new ApolloClient({
    uri:'/graphql,
});


import React from 'react';
import { render } from 'react-dom';
import { ApolloProvider } from '@apollo/react-hooks';

const App = () => (
  <ApolloProvider client={client}>
    <div>
      <h2>My first Apollo app</h2>
    </div>
  </ApolloProvider>
);

render(<App />, document.getElementById('root'));
```

Apollo client links

```
import ...

const httpLink = createHttpLink({
    uri: `/graphql`,
});

let token: string = ''
const service = new AuthService();

const authLink = setContext((_, { headers }) => {
    if (token) {
        return {
            headers: {
                ...headers,
                authorization: `Bearer ${token}`,
            },
        };
    } else {
        // get token
    }
});
```

```
const link = ApolloLink.from([retryLink, authLink,
errorLink, httpLink]);

const cache = new InMemoryCache({ fragmentMatcher
});

const client = new ApolloClient({
    link,
    cache,
    connectToDevTools: true,
});
```

Query

```
import gql from 'graphql-tag';

const getComments = gql`
    query comments($objectIds: [Int], $objectType: Int) {
        comments(objectIds: $objectIds, objectType: $objectType) {
            id
            text
            employeeLogin
            dateTimeAdded
            isMyComment
            objectId
            commentEmployee {
                number
                firstName
                lastName
                middleName
                login
            }
        }
    }
`;
export default getComments;
```

Query

```
import React, { useState, FC } from 'react';
import { useQuery } from 'react-apollo-hooks';
import getComments from '../graphql/queries/getComments';

const CommentsContainer: FC<Props> = ({ objectIds, objectType }) => {
    const { data, loading, error } = useQuery(getComments, {
        variables: {
            objectIds,
            objectType,
        },
    });
    // onDelete

    if (loading) {
        return <Icon design='spinner' position='absolute' />;
    }

    if (error) {
        return <Error />;
    }

    return <Comments comments={data.comments} onDelete={onDelete} />;
};
```

Multiple query

```
const getLeftSidebarInfo = gql`
    query getLeftSidebarInfo {
        getSelfInfo {
            id
            fullName
            email
            employeeNumber
            login
            leader
        }

        budgetAvailableTeams {
            id
            name
        }

        budgetAvailableDepartments {
            id
            name
        }
}
`;
```

Mutation

```
import gql from 'graphql-tag';

const deleteComment = gql`
    mutation deleteComment($commentId: Int) {
        deleteComment(commentId: $commentId)
    }
`;
export default deleteComment;

const deleteCommentMutation = useMutation(deleteComment);

const onDelete = (id) => {
      if (!id) {
          return;
      }

      deleteCommentMutation({
          variables: {
              commentId: id,
          },
          refetchQueries: [{ query: getComments, variables: { objectIds, objectType } }],
      });
};
```

Generate types

```
/* tslint:disable */
/* eslint-disable */
// This file was automatically generated and should not be edited.

// ====================================================
// GraphQL query operation: comments
// ====================================================

export interface comments_comments_commentEmployee {
    __typename: 'CommentEmployee';
    number: string | null;
    firstName: string | null;
    lastName: string | null;
    middleName: string | null;
    login: string;
}

export interface comments_comments {
    __typename: 'Comment';
    id: number;
    text: string | null;
    employeeLogin: string;
    dateTimeAdded: any | null;
    isMyComment: boolean;
    objectId: number;
    commentEmployee: comments_comments_commentEmployee | null;
}
```

Interacting with cached data

```
const query = gql`
  query MyTodoAppQuery {
    todos {
      id
      text
      completed
    }
  }
`;

// Get the current to-do list
const data = client.readQuery({ query });

const myNewTodo = {
  id: '6',
  text: 'Start using Apollo Client.',
  completed: false,
  __typename: 'Todo',
};

// Write back to the to-do list and include the new item
client.writeQuery({
  query,
  data: {
    todos: [...data.todos, myNewTodo],
  },
});
```

Testing

```
import { MockedProvider } from '@apollo/react-testing';

// The component AND the query need to be exported
import { GET_DOG_QUERY, Dog } from './dog';
const mocks = [
  {
    request: {
      query: GET_DOG_QUERY,
      variables: {
        name: 'Buck',
      },
    },
    result: {
      data: {
        dog: { id: '1', name: 'Buck', breed: 'bulldog' },
      },
    },
  },
];

it('renders without error', () => {
  renderer.create(
    <MockedProvider mocks={mocks} addTypename={false}>
      <Dog name="Buck" />
    </MockedProvider>,
  );
});
```

Райффайзен БАНК

Спасибо за внимание!