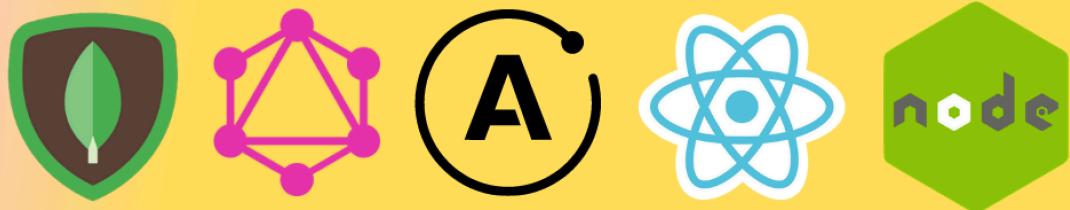




# Build a Collaborative ToDo App Using M.E.R.N. Stack (for beginners)

GraphQL API using NodeJS and MongoDB

A screenshot of a mobile application interface. At the top, there's a header bar with three user profile icons, a search icon, and a more options icon. The main content area has a dark background. It shows a list titled "Collaborative ToDo App (MERN stack)" with three items. The first item is "Prerequisites" with two bullet points: "NodeJS: <https://nodejs.org/en/download/>" and "NPM (comes with NodeJS)". The second item is "Backend (GraphQL API with NodeJS and MongoDB)" with four sub-items: "Bootstrap the project" (checkbox, checked, by user "vadim97"), "Follow the getting started guide: <https://www.apollographql.com/docs/apollo-server/getting-started/>" (checkbox, checked, by user "vadimsavin" on Apr 2), "Execute your first query" (checkbox, unchecked), and "Setup Nodemon to restart the server on changes" (checkbox, unchecked). A small circular badge with the number "1" is visible at the bottom right of the list area.

## In this video

- Setup a NodeJS project
- |||
- |||
- |||
- |||
- |||
- |||
- |||



## In this video

- Setup a NodeJS project
- Create a “Hello world” GraphQL API



## In this video

- Setup a NodeJS project
- Create a “Hello world” GraphQL API
- Configure the Database



## In this video

- ||| - Setup a NodeJS project
- ||| - Create a “Hello world” GraphQL API
- ||| - Configure the Database
- ||| - Write the Schema definition
- |||
- |||
- |||
- |||



## In this video

- ||| - Setup a NodeJS project
- ||| - Create a “Hello world” GraphQL API
- ||| - Configure the Database
- ||| - Write the Schema definition
- ||| - Implement Queries and Mutations



## In this video

- ||| - Setup a NodeJS project
- ||| - Create a “Hello world” GraphQL API
- ||| - Configure the Database
- ||| - Write the Schema definition
- ||| - Implement Queries and Mutations
- ||| - Authentication



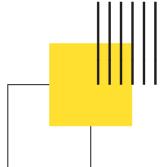
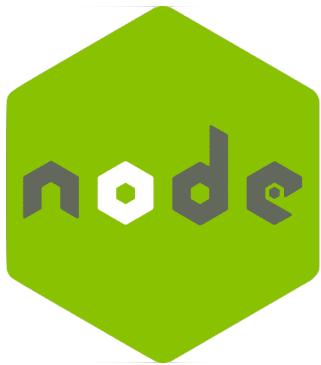
## In this video

- ||| - Setup a NodeJS project
- ||| - Create a “Hello world” GraphQL API
- ||| - Configure the Database
- ||| - Write the Schema definition
- ||| - Implement Queries and Mutations
- ||| - Authentication
- ||| - Best Practices



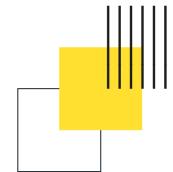
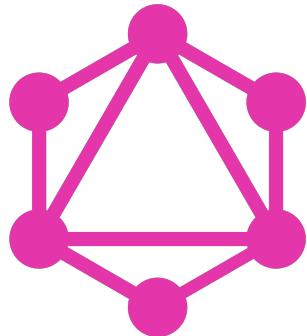
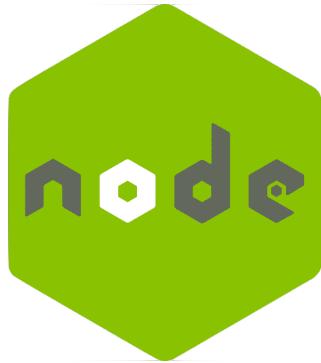
# Backend technologies

---



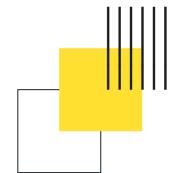
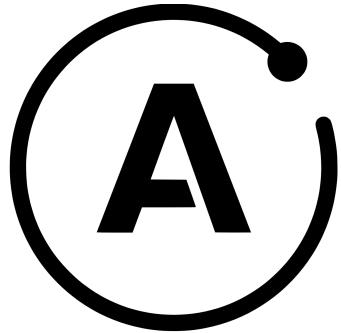
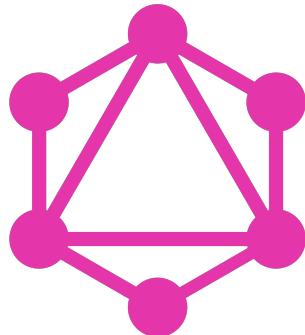
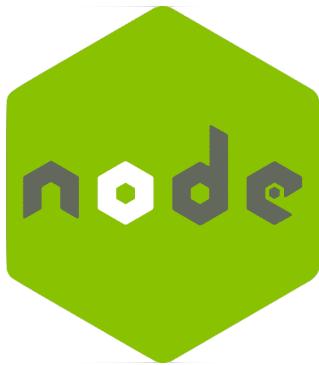
# Backend technologies

---



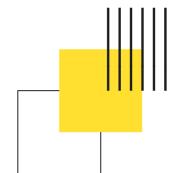
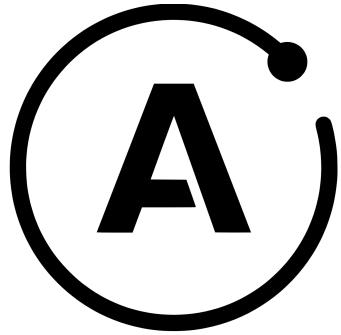
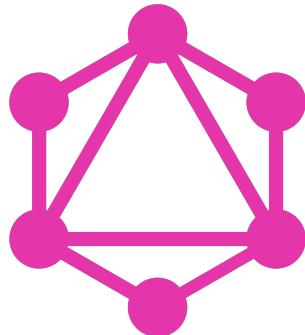
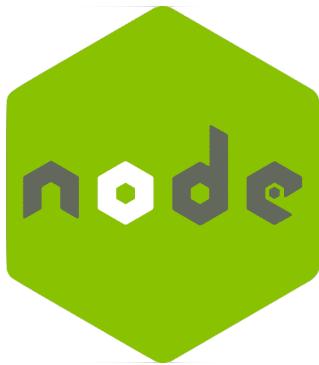
# Backend technologies

---



# Backend technologies

---



# notJust

//development

## Who am I



- Full stack developer for >7yrs
- CTO @Fitinium
- ex Amazon SDE
- Passionate about coding and building impactful startups.

I AM notJust  
//developer

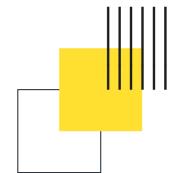
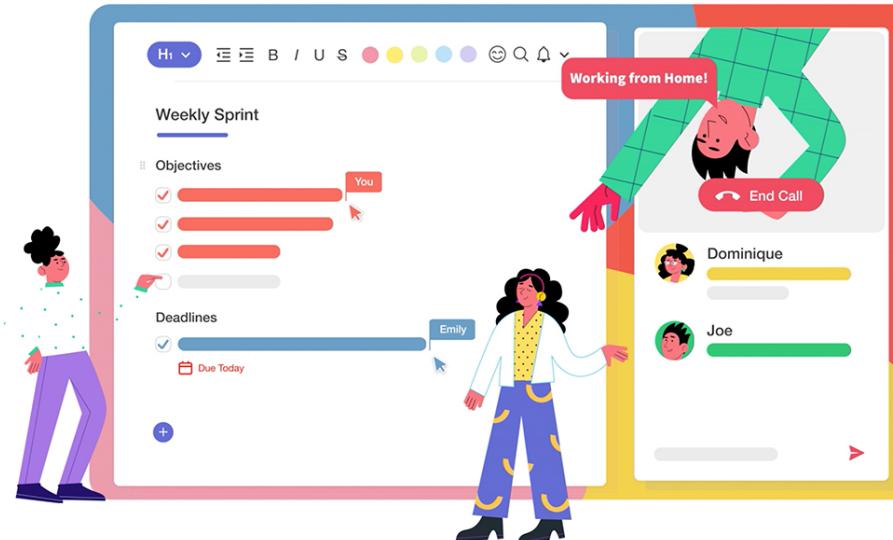


# The sponsor of the Video

---



# taskade



## Prerequisites

1. Asset Bundle (dummy data, images, icons, PDF presentation):

||| <https://assets.notjust.dev/taskade>

||

2. Taskade Board for Project Management

|| <https://bit.ly/3rLJ694>

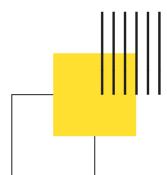
||

|||

## What is GraphQL

---

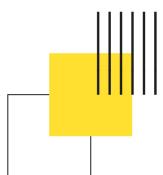
*GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.*



## What is GraphQL

---

*GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.*



# The Schema Definition Language

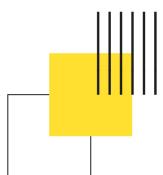
```
type User {  
    id: ID!  
    name: String!  
    posts: [Post!]!  
}  
  
type Post {  
    id: ID!  
    title: String!  
    author: User!  
}  
  
type Query {  
    user(id: ID!): User  
    feed: [Post!]!  
}  
  
type Mutation {  
    createUser(name: String!): User  
    writePost: (  
        title: String!,  
        authorId: ID!  
    )  
}
```



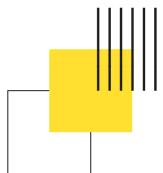
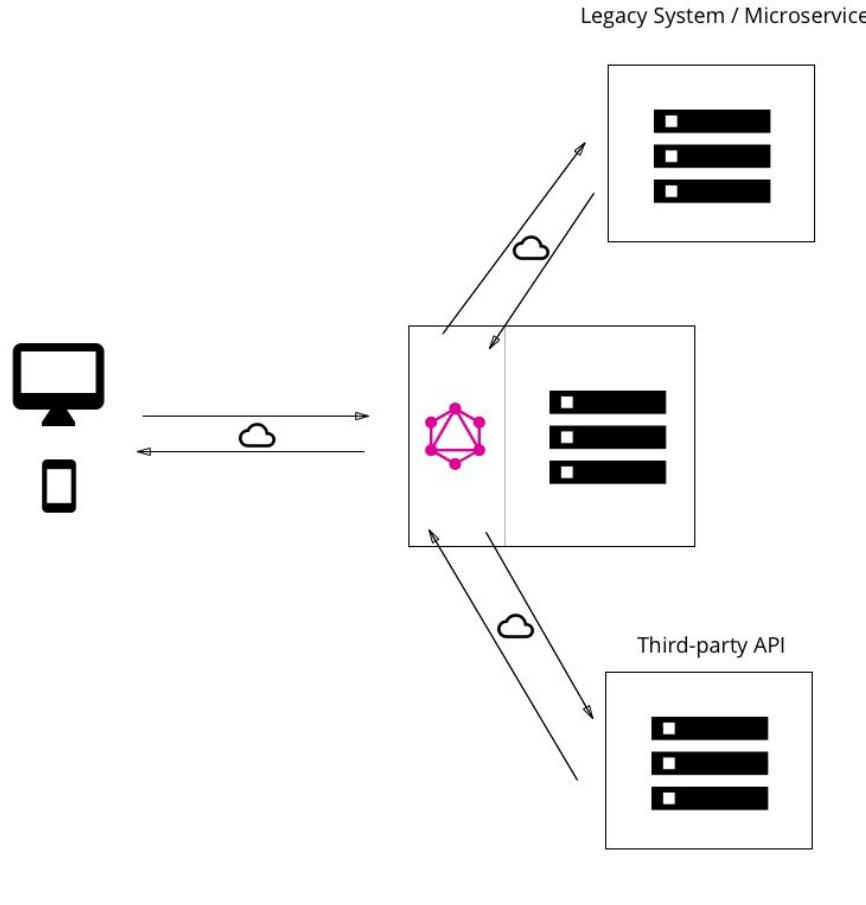
## What is GraphQL

---

*GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.*



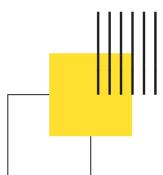
# What is GraphQL

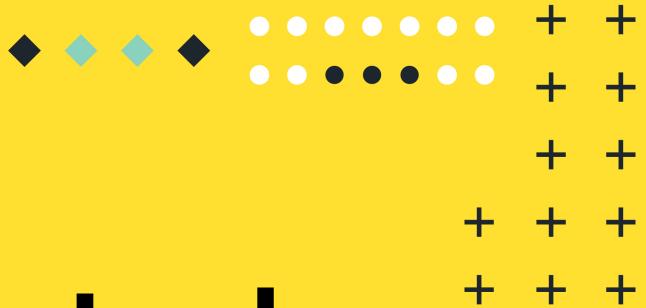


## What is GraphQL

---

*GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and understandable description of the data in your API, gives clients the power to ask for exactly what they need and nothing more, makes it easier to evolve APIs over time, and enables powerful developer tools.*





# Let's get started



<https://bit.ly/3rLJ694>

