Trends II

Low-code application development

Atilla Balin, Adam Lenez, Kim Mignon and Aslan Jousoupkhadzjiev

31-1-2022

Inhoud

Introduction

For this group project, we chose to work on the IT-trend called “low-code programming”, which entails the graphic development of software applications by making use of a graphic user interface (GUI). As proof-of-concept (PoC) of this IT-trend, we developed a movie database website with the free low-code platform teleportHQ. On this movie database website, users can look up different movies and TV shows and check the trailer, the audience score and other relevant information about the selected movie/TV show. To get the required data of all the movies, an API connection was made with The Movie Database API service. In this way, our website is connected with this database, and it can so retrieve all the requested data about the movies. Finally, to launch our website, we made use of GitHub actions to automate the process of launching the website, and possible changes to it, directly onto the Heroku hosting platform.

The goal of this project is to show the power of low-code programming and its value in the IT-world. What are the advantages and disadvantages of using this ‘way’ of developing software applications and for what applications is low-code programming best used? With the development of our website, we hope to show the strong and weak points of low-code programming and its usefulness in businesses.

In this report, we discuss the low-code programming trend and its relevance in the IT-world. We further describe the different techniques used for the POC and the progress of making the website with our personal remarks. In the end, we summarise and conclude our findings on the low-code programming trend.

1. Low-code programming

1.1 The concept

Low-code programming is the graphic development of software without writing lines of code for it. Applications are built visually by making use of a GUI that provides basic logic and drag-and-drop tools. In this way, users with no advanced knowledge of coding are capable to create software applications just by using the techniques provided by a visual interface. This way of building software also increases the development speed as developers don't have to take boiler-plate code into account so they can fully focus on the important functionalities of the application. In essence, low-code programming is a simplified way of software development.

However, to visually build an application, a low-code platform is needed to translate everything into code and to configurate the application. Low-code platforms provide a development environment in which users can freely develop an application through a GUI that is equipped with visual modelling features, of which the drag-and-drop functionality is possibly the most crucial. The platform enables the user to incorporate building blocks into workflows and apps and it takes the complexity of programming out of the hands of the developers.

Some platforms can produce fully operational applications, but most require additional coding by the developer, yet this also strongly depends on the complexity and type of the application itself. The requirement of additional coding can be seen as a flaw of the low-code platforms as it makes the platforms less accessible for users with poor coding knowledge that cannot manually edit code. However, for skilled users the adaptability of code can be seen as an advantage as it gives them the same possibilities and flexibility as manual coding while still being able to work faster via the GUI.

Low-code platforms can be divided in different categories depending on what the required application is. Forrester (a company in business process management) categorized all low-code platforms into five different categories, with the platforms of each category specialized in that specific task. The five categories are: general purpose, request handling, process, mobile first and database. A lot of platforms are already available, from which some are (luckily) open-source. It is predicted that by 2023 about 50 % of companies will use low-code programming for the development of applications. In the future, low-code will thus become a key factor for businesses.

1.2 History

A breakthrough came with the invention of FORTRAN by computer giant, IBM. Invented in the 1950s, this was the first revolutionary and incredibly popular programming language. But FORTRAN still had a problem. It was mostly useful for scientific and numeric computing. Also, it wasn’t as intuitive as people needed it to be. In addition to all of this, it was limited in what it could do.

And that’s where COBOL came in to pick up the slack. It wasn’t targeted at scientists and mathematicians looking to find the secrets of the universe. COBOL helped them find solutions to business tasks. There was also the added benefit that COBOL supported object-oriented programming, something we take for granted with languages like C++ and Java.

Inspired by COBOL, other languages popped up, providing small improvements and ease of use.

C was a radical departure from the likes of COBOL, FORTRAN, and other languages of its time. It was structured, written with English syntax, and usable for a variety of applications. Over the years, it’s been followed up by C++, which added object-oriented programming concepts into it like inheritance, encapsulation, and polymorphism.

The biggest change after C++ came by way of C#, which was better equipped to create web applications. After the explosive growth of the Internet, that was the biggest motivator for the advancement of programming languages, like Java, Python, PHP, and more.

The Internet is a huge platform where different systems running different platforms all must work well with each other. Because of that, programming languages had to evolve to support that need.

Web applications became more popular, browsers became more complex, we started using smaller and simpler scripts for simpler tasks, and instead of having a complete programming language, there was a focus on function. If a language could do one thing, and do it well, it was useful. Otherwise, it was thrown to the waste pile. Applications needed to be developed at a faster pace, and languages had to be easy enough to support that.

This is the time when concepts like rapid application development and low code started. Modern languages were focused on helping developers speed up the development process, not spend hours searching for that one semicolon they missed. Low-code development platforms emerged in the 1990s and early 2000s as a result of this drive for rapid application development. Since then their abundancy in companies only grew.

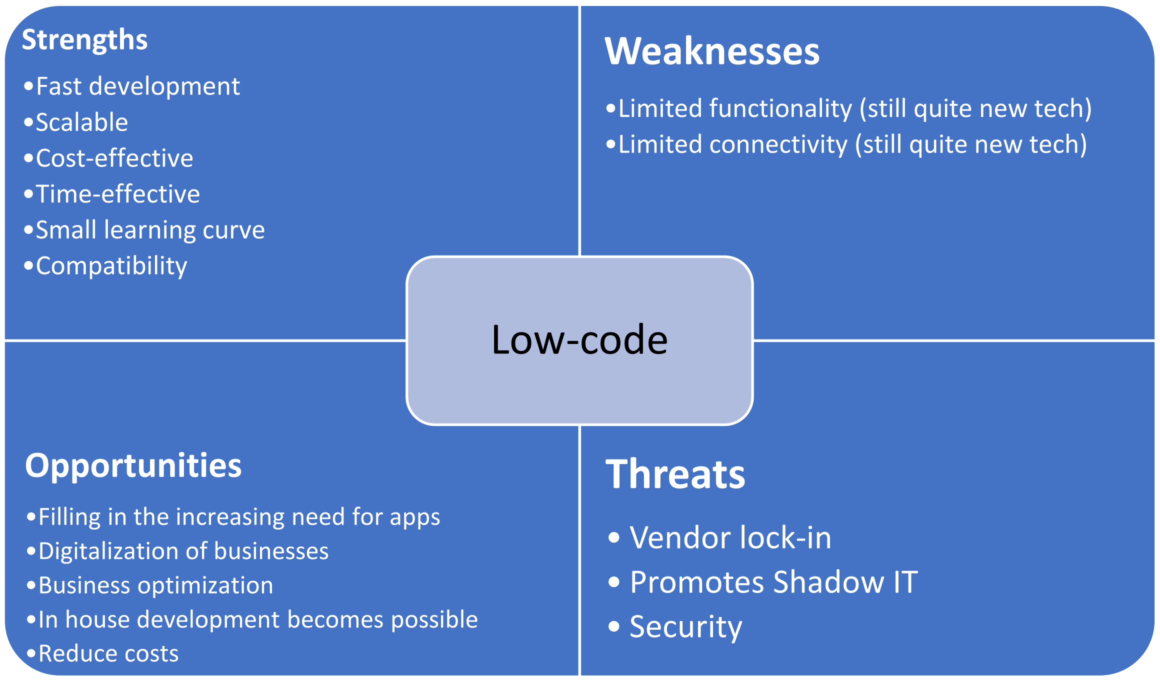
1.3 Relevance in IT

The digitalization of the business processes and procedures have led to increased demand for applications. Any business process you can possibly imagine could be facilitated with an application. The COVID19 pandemic has increased the demand for applications even further. Suddenly companies introduced desk reservations at the office to make sure the social distancing rules can be respected or adapted with an automated workflow to make it possible to submit and approve expenses of the employees.

Normally this kind of application development would be very expensive and time consuming. Thankfully there are low-code application building service providers who have seen this increasing demand and found a way to fill it in. Those services allow anyone with basic IT knowledge to build applications. However, it is not meant for beginners only. Also experienced developers can leverage the low-code application building in order to create simple applications as quick as possible, meaning that the application building process can go a lot more efficient.

This new way of application building was at first received with a lot of scepsis but as it proved itself to be very powerful and very agile, more and more companies have started to embrace it.

SWOT analysis



Strengths

Fast development:	Creating a simple application based on existing data, for example in excel, can be a matter of minutes.
Scalable:	Very often, this kind of application is deployed on a cloud so they can be used by many users at the same time and can handle a lot of data.
Cost-effective:	You no longer need to invest in expensive hardware. Usually, the service provider offers the computing power and storage in the cloud
Time-effective:	As the amount of code required to be written is quite limited and low-code provides you with a set of prebuilt templates and elements, building an application can be very quick.
Small learning curve:	Low code also reduces the learning curve for training new people to maintain and modify the code.
Compatibility:	Low-code mobile application development allows for cross-platform mobility. That means you no longer have to spend more time developing your application for a specific platform and can instead focus on making the app better overall.

Weaknesses

Limited functionality:	It is possible you won't be able to create super complex applications, but those services are continuously evolving and introducing new features frequently.
Limited connectivity:	Sometimes you might face difficulties connecting your low-code applications to an existing enterprise suite due to lack of connectivity possibilities. Yet again, this might change over time as low-code is constantly evolving.

Opportunities

Filling in the increasing need for apps:	More people are able to create apps, thus creating a bigger growth of new coming apps.
Digitalization of businesses:	Low code platforms can simplify key challenges for in-house IT teams. Low code platform features such as electronic forms and process automation have assisted organizations in overcoming difficulties in a significantly shorter period of time.
Business optimization:	Developers can start working on their mobile application without getting bogged down in the basic developmental ways of coding.
In house development becomes possible:	This resulting in a smoother operation between the idea and end product.
Reduce costs:	Low code mobile development platforms offer a speedier execution when it comes to developing a mobile app, which results in a lower overall cost.

Threats

Vendor lock-in:	Some generate convoluted code that's nearly impossible to maintain outside of the platform. Others won't let you edit your applications once you stop using the tool.
Promote Shadow IT:	Multiple IT leaders have discovered that departments within their organization had adopted a low-code tool without IT's knowledge. Not only does this waste resources, it creates security risks.
Security:	The platform itself may write unsecure code or create opportunities of scale for hackers. (When creating multiple apps on same platform)

2 Proof-of-concept: Movie database website

2.1 Idea/Goal

As proof-of-concept of the functionalities of low-code programming, we aimed to develop a movie database website using a low-code programming platform. To do so, we needed a platform that is free and that has GitHub integration so we could submit the different versions of our project in a GitHub repository. The platform teleportHQ meets these two requirements, and we chose to use this platform to build the static website. On the movie database website, we wanted users to be able to search for movies and TV-shows and to get detailed information about a selected movie/TV show. To get all this information, we needed to integrate an API connection in our code in order to connect our website to a database containing all the required data about the movies/shows. Finally, to launch our website automatically on a hosting platform, we made use of the GitHub Actions functionalities and the Heroku hosting platform.

In the following paragraphs we describe the development process of our PoC based on the different techniques (low-code programming, API, GitHub Actions and Heroku) we had to use.

2.2 TeleportHQ

2.2.1 What is it?

As already mentioned, we chose to develop our website with the teleportHQ platform, because this platform is free and provides integration with GitHub. TeleportHQ is a free-to-use, all-in-one platform for the front-end development of static websites. Through an interface, it provides simple drag-and-drop elements to build a page and adjust the design (Fig. 1). However, for the design, users can also choose different website templates from teleportHQ itself.

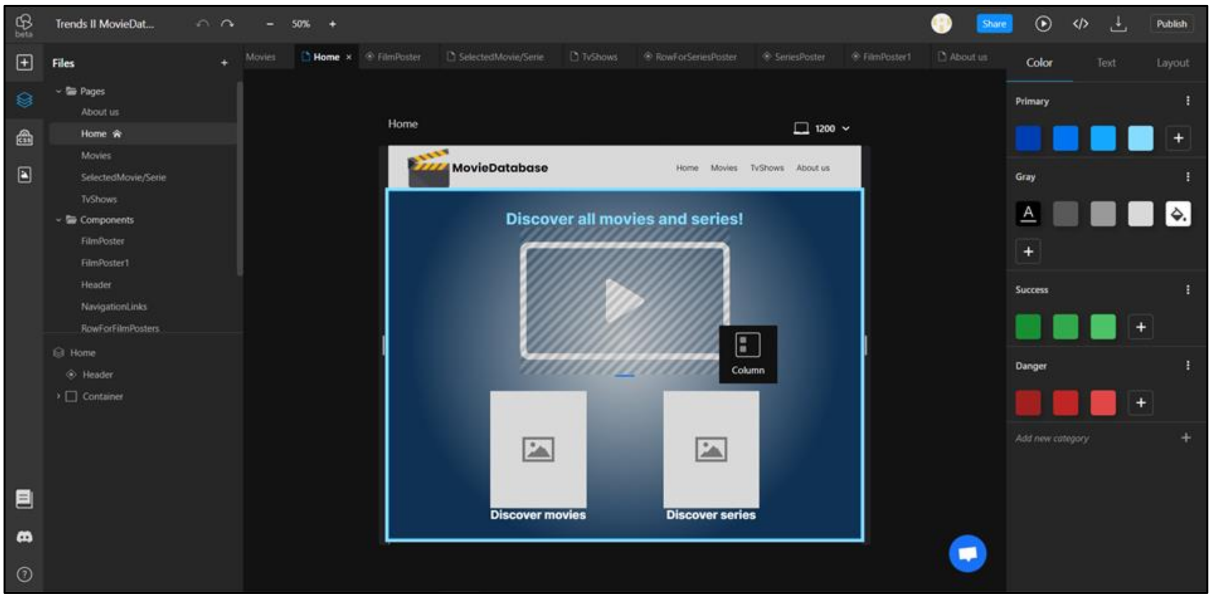
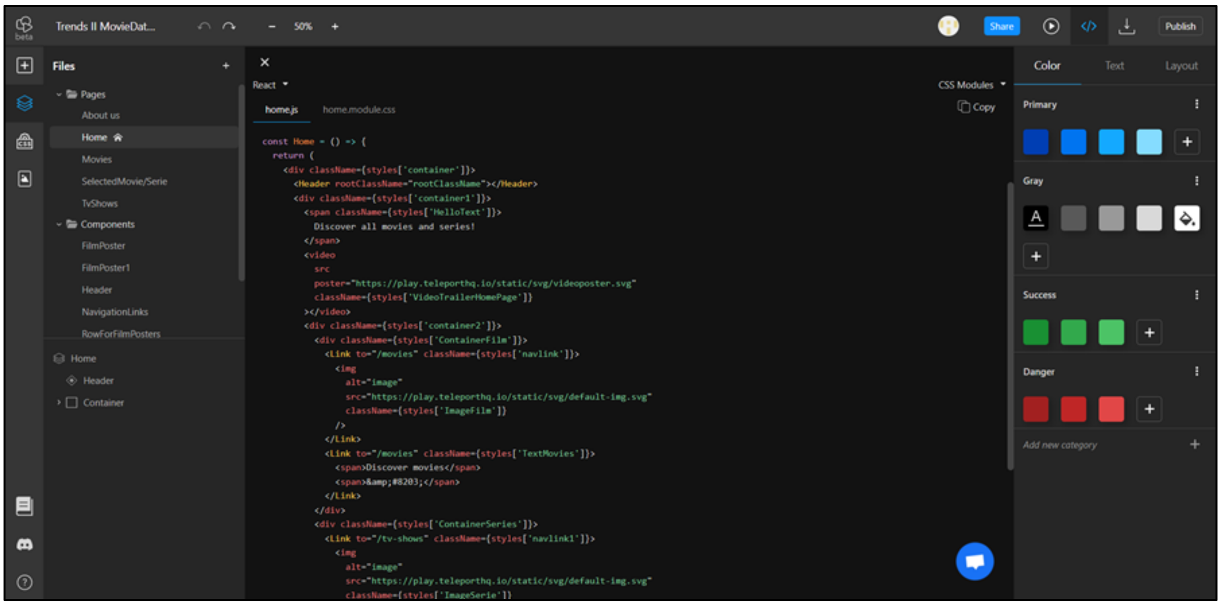


Figure 1: GUI of teleportHQ: Users can select different elements, layers and components on the left column, and they can drag and drop these elements on the page in the middle work aera. The right column shows the visual and advanced properties of each element which can be adjusted by the user.

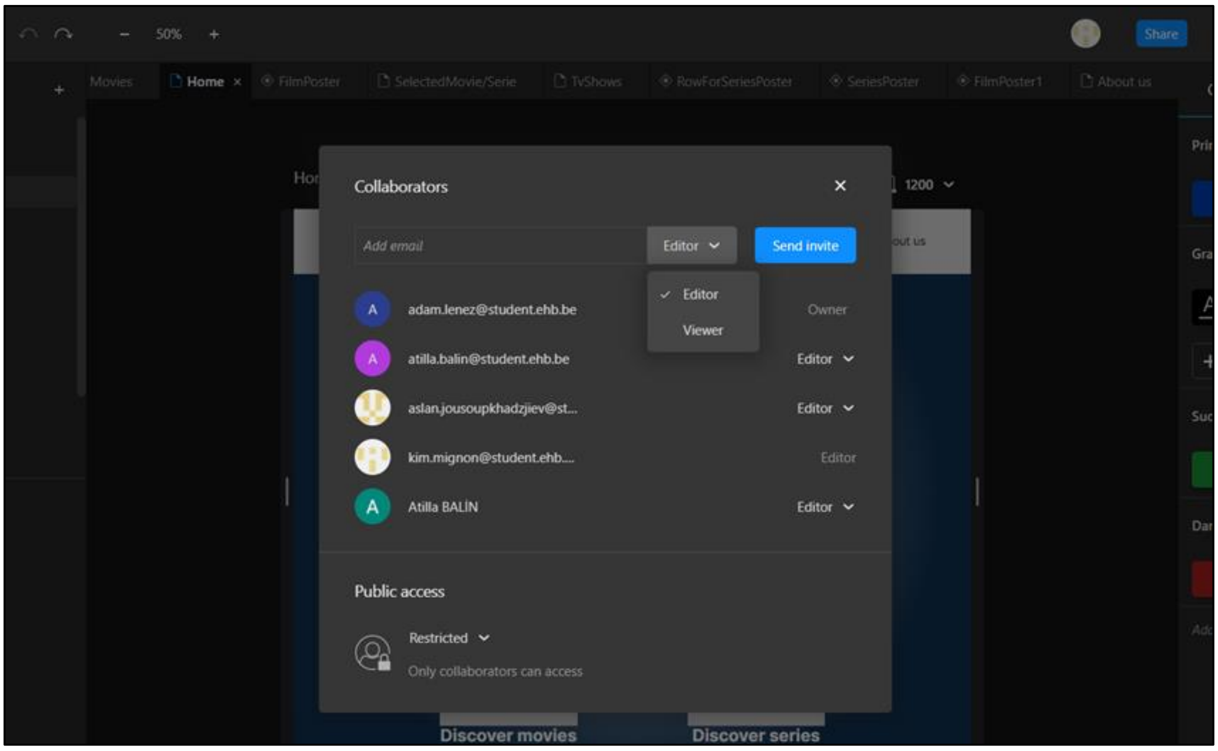
Websites can be hosted free on the teleportHQ domain (via the Publish button) or on a domain of choice. The underlying code can be viewed with the "generated code" button (</>) and is written in in JavaScript (JS) and CSS (Fig. 2). According to the teleportHQ website, clean code can be exported to HTML, CSS and JS files.

Users are able to invite other users to work on a project. Via the "Share" button, an email invite can be sent to other collaborators which can be selected as an editor or just as a viewer (Fig. 3). A collaborator has to accept the email invite and receives a link to the workspace of the corresponding project. This is what we also did for building our project: one of us made a new project in teleportHQ and invited the other group members so we could all work on the static website.



```
const Home = () => {
  return (
    <div className={styles['container']}>
      <Header rootClassName="rootClassName"></Header>
      <div className={styles['container1']}>
        <span className={styles['HelloText']}>
          Discover all movies and series!
        </span>
        <video
          src
          poster="https://play.teleporthq.io/static/svg/videoposter.svg"
          className={styles['VideoTrailerHomePage']}
        ></video>
        <div className={styles['container2']}>
          <div className={styles['ContainerFilm']}>
            <Link to="/movies" className={styles['navlink']}>
              <img
                alt="image"
                src="https://play.teleporthq.io/static/svg/default-img.svg"
                className={styles['ImageFilm']}
              />
            </Link>
            <Link to="/movies" className={styles['TextMovies']}>
              <span>Discover movies</span>
              <span>&amp;#8203;</span>
            </Link>
          </div>
          <div className={styles['ContainerSeries']}>
            <Link to="/tv-shows" className={styles['navlink1']}>
              <img
                alt="image"
                src="https://play.teleporthq.io/static/svg/default-img.svg"
                className={styles['ImageSerie']}
```

Figure 2: The underlying code of each page can be viewed. The code is written is in .js and .css.



Collaborators

Add email | Editor | Send invite

✓ Editor
Viewer

adam.lenez@student.ehb.be	Owner
atilla.balin@student.ehb.be	Editor
aslan.jousoupkhadzjiev@st...	Editor
kim.mignon@student.ehb....	Editor
Atilla BALİN	Editor

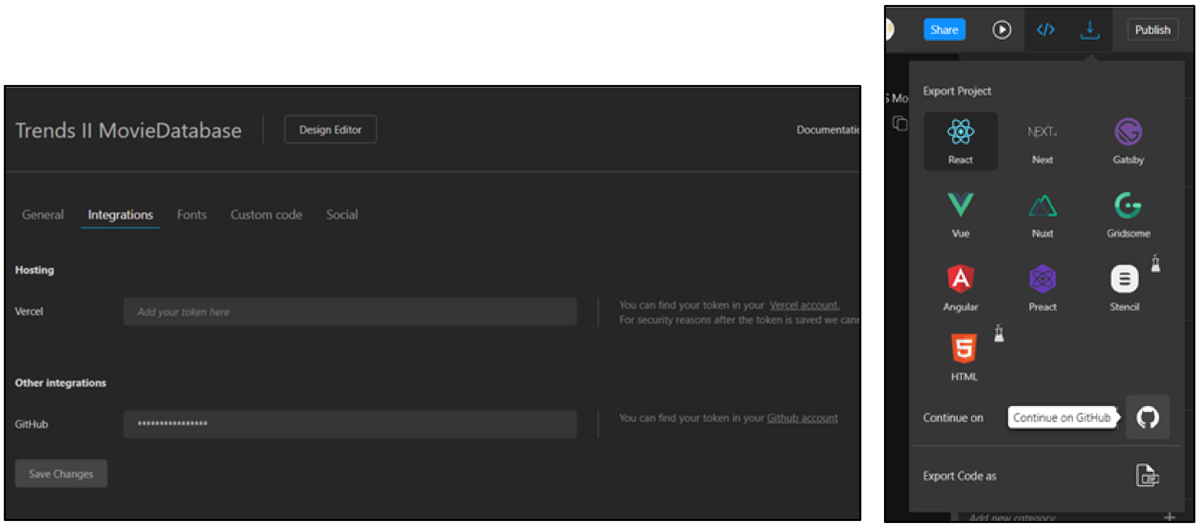
Public access

Restricted
Only collaborators can access

Figure 3: Collaborators can be invited via an email invite. They can be selected as an editor or just as a viewer.

Finally, a user can integrate a project in teleportHQ with GitHub by entering his/her personal GitHub token (Fig 4 left). Hereafter, the user is able the export the project to GitHub via the "export" dropdown menu (Fig 4 right). When the first commit to GitHub is made, a public repository will be automatically created for the project and from that moment, all commits will be sent to that repository. The repository is owned by the GitHub-account corresponding to the GitHub token that was entered.

This user can then share the repository with other GitHub-accounts and in this way, other users can be co-owner of the repository. Again, this is what we did for our project: one group member entered his token and then shared the automatically created repository with the GitHub-accounts of the other group members. From that moment, all group members had full control of the project as they could work on the website and made commits in the teleportHQ workspace and they were also able to pull data directly from the GitHub repository and make changes and new commits.



You can find your token in your Vercel account.
For security reasons after the token is saved we can

You can find your token in your Github account

Figure 4: Left) To integrate a teleportHQ token with GitHub a personal Github-Token has to be entered. Right) When this is done, the commits to GitHub can be made with the dropdown menu of the export button. With the first commit, a repository is automatically created.

On big disadvantage of the integration of teleportHQ with GitHub is that no adjusted message could be added to the commit. The same message "Commit from TeleportHQ" is always added, which means that there is less clarity about what is finished in each commit.

2.2.2 *Making of static website*

Our movie database website exists of two main sections: a movie section and a TV-Show section. These two sections are linked via a home page. From this home page, users can choose to look-up movies or TV-shows by clicking on the “Discover movies” or “Discover series” button respectively (Fig. 5). When a user selects one of these two buttons, they our directed to either a “Movies” page or a “TV-Shows” page which displays all the movies or TV-shows in alphabetical order. When a user eventually selects a specific movie/TV-Show, the user is directed to a new page displaying the information about that movie/TV-show.

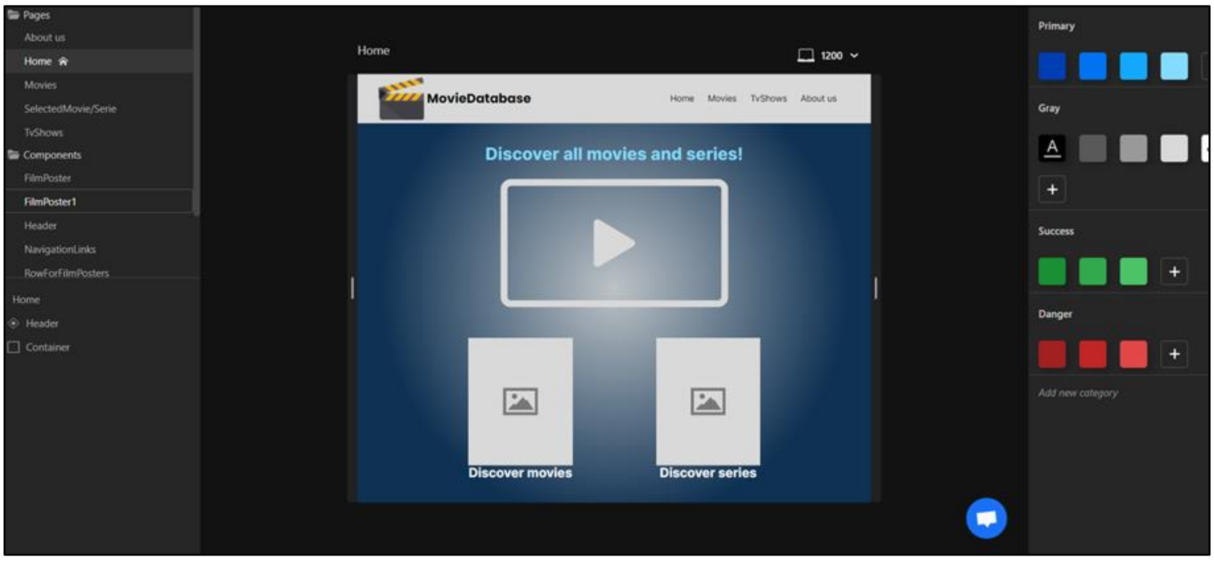


Figure 5: The home page how it is created in teleportHQ, showing the different elements and components. The photos and underlying texts “Discover movies” and “Discover series” are used to redirect the user to other pages.

This means that four different pages had to be created, namely a homepage, a “Movies”-page, a “TV Shows”-page and a page for the selected movie/TV-show. As we also wanted to include some personal information about ourselves on the website, we made a sixth page called “About us”. Pages (and components, see further) are created in teleportHQ by going to the “Files” sidebar and pressing the plus-sign (Fig. 6 left).

Links between pages are easily made in teleportHQ: users just have to click on the element that they want to make as a link and then click on the paperclip-like symbol. Next, users can enter a URL or select one of the pages to which they want to make a link (Fig. 6 right). A link can be made with all types of elements (not only buttons).

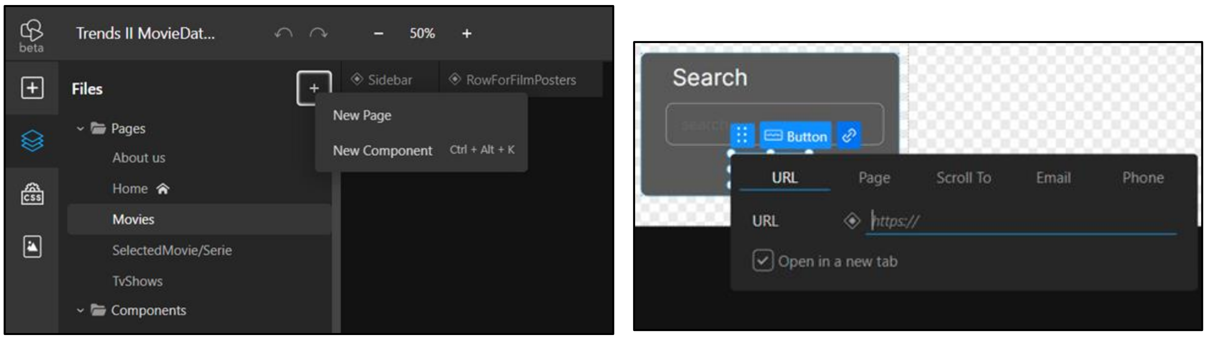


Figure 6: Creating new pages and components in teleportHQ and making links to other pages.

After creating the pages, different components were made. Components are elements that consist of a vast set of other elements. In this way, that vast set of elements can be added to a page as a whole, in one component. For our website, we made a header, a side bar and a poster component. In Fig. 7, the side bar component is shown as an example. A component is created by dragging different elements from the "elements" column that is left on the interface. The design, position and other visuals of the elements that ensemble the component can be changed in the right column of the interface. After creating the three components, each page received a header and the "Movies" and "TV shows" pages also received a side bar so that the users could order movies/TV-shows by name, popularity, rating or select them by name.

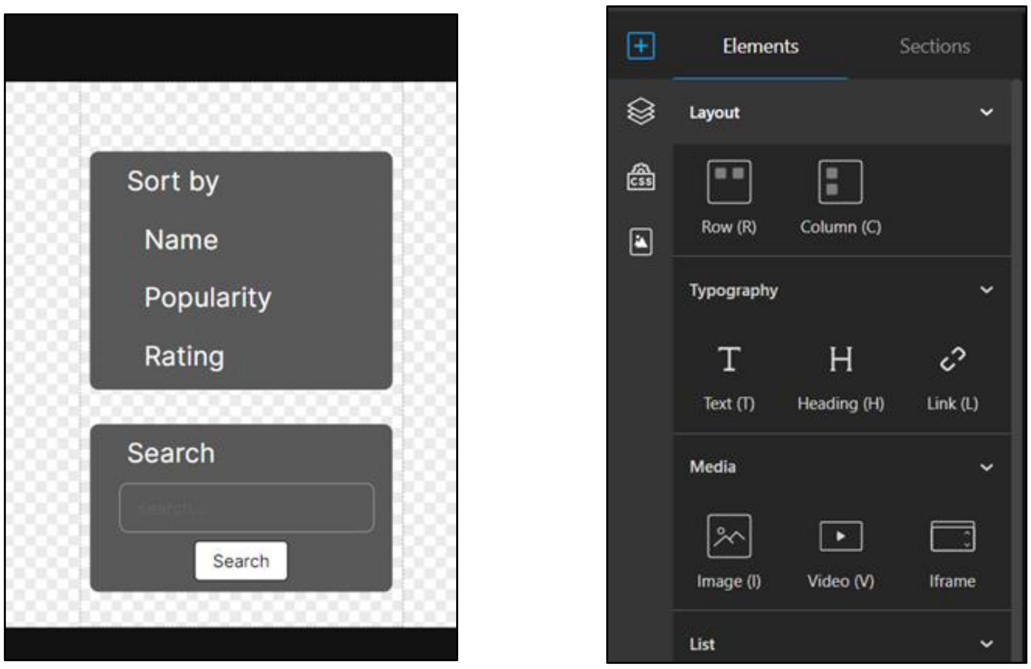


Figure 7: Example of the side bar component. Components are ensembled of different elements chosen from the Elements side bar left on the interface.

Finally, we wanted to fill the “Movies” and “TV shows” pages with posters of the movies and TV-shows respectively. The number of posters that has to be shown on these pages is dependent on the search request of the users: sometimes this can be a large number of posters and other times just a few. Therefore, we wanted the page to be dynamically filled with posters corresponding to the search result. For this, we added a container element in the corresponding pages, in which new posters can be added.

One big disadvantages of the container elements in teleportHQ is that either a column element or a row element can be chosen and not a combination of both. This means that, when we choose for example a column element, all posters will be added underneath each other with rows of only one poster. When we would choose a row element on the other hand, all posters will be displayed next to each other. We tried to avoid this problem by making a new component ensembled of a row element filled with three poster components (= ”series”-component). The plan was then to dynamically fill the column element on the “Movies” and “TV Shows” page with this new “series”-components to have at least rows of three posters displayed on the page. Unfortunately, we could not use this set-up as it was not possible for us (when making the dynamic code) to first create new posters with the movie data, then to create new “series”-components filled with three posters and then to add all the created “series”-components in the column container. We therefore added only one column element to each of the two pages and we then had to manually adjust the CSS code for this element afterwards so that the posters could be added one by one in rows of four posters.

During the process of making this static website in teleportHQ, we encountered some other disadvantages, but also important advantages of the platform. Here is a small list of our findings:

Disadvantages:

- Only static website (although it is the concept), coding knowledge is required to make the site dynamic.
- Git integration only consists of Push-command without adjustable messages.
- Limited possibilities of elements (e.g. row or column elements or certain details not possible like showing only one side of a button)
- In case of bad network, the synchronisation sometimes fails.

Advantages:

- Easy to use from the start, very intuitive.
- Easy to use together with other users (if network is okay), you can see what other users are working on at the moment with tags.
- Built-in templates available and easy to select.
- The concept of the components is very helpful and it makes it possible to use consistent elements on different pages.

2.3 API connection WITH Node.js (stage 2)

2.3.1 What is API?

After making the static website, different components had to be filled with data about the movies or TV-Shows. To get this data, we used an API. API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

When used in the context of web development, an API is typically defined as a set of specifications, such as Hypertext Transfer Protocol (HTTP) request messages, along with a definition of the structure of response messages, usually in an Extensible Markup Language (XML) or JavaScript Object Notation (JSON) format.

Web APIs allow the combination of multiple APIs into new applications known as mashups. In the social media space, web APIs have allowed web communities to facilitate sharing content and data between communities and applications. In this way, content that is created in one place dynamically can be posted and updated to multiple locations on the web. For example, Twitter's REST API allows developers to access core Twitter data and the Search API provides methods for developers to interact with Twitter Search and trends data.

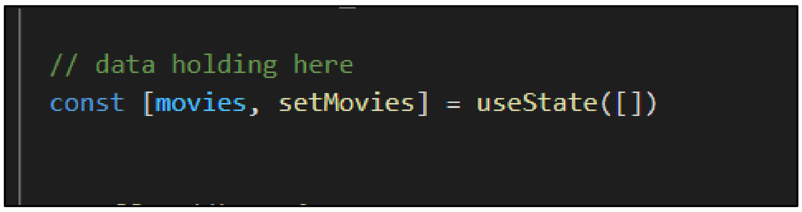
2.3.2 *Integrating API with react.js*

Steps taken

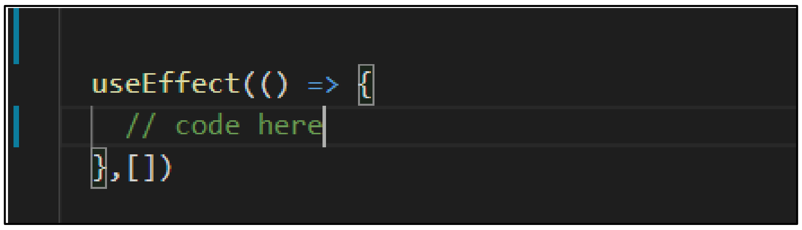
For integrating an API we made use of Node.js. Node.js is a JavaScript framework/environment with which users can run JavaScript code outside a webserver and instead run it locally. In this way, developers can use JavaScript to write command-line tools and to produce dynamic web page content before the page is sent to the web browser. Node.js is generally used for the development of server-site applications and back-end programs.

To simplify the retrieval of data from the API, we used the React library which is a JavaScript library for building user interfaces. React.js enabled us to use Hooks (useState and UseEffect) with which users can extract stateful logic from a component so it can be tested independently and reused. This is what we used on the API database.

We started by importing the useState and useEffect hooks in the movies.js file (top-level-component) (Fig. 8). The userState hook is for sorting the data and by using the useEffect hook users tell react.js that your component needs to do something after rendering (e.g. fetching data).



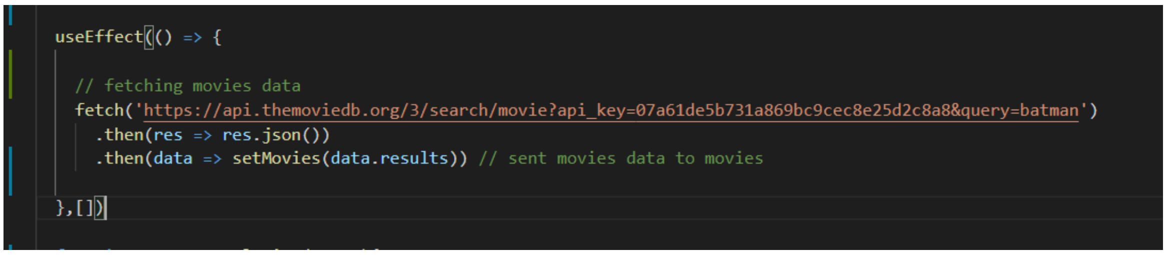
```
// data holding here
const [movies, setMovies] = useState([])
```



```
useEffect(() => {
  // code here
},[])
```

Figure 8: use of the useState and useEffect hooks.

Next, the fetchAPI-command was used to ask for the http (GET) request form ”`https://api.themoviedb.org/3/search/movie?api_key=07a61de5b731a869bc9cec8e25d2c8a8&query=batman`” to fetch JSON data in useEffect() (Fig. 9).



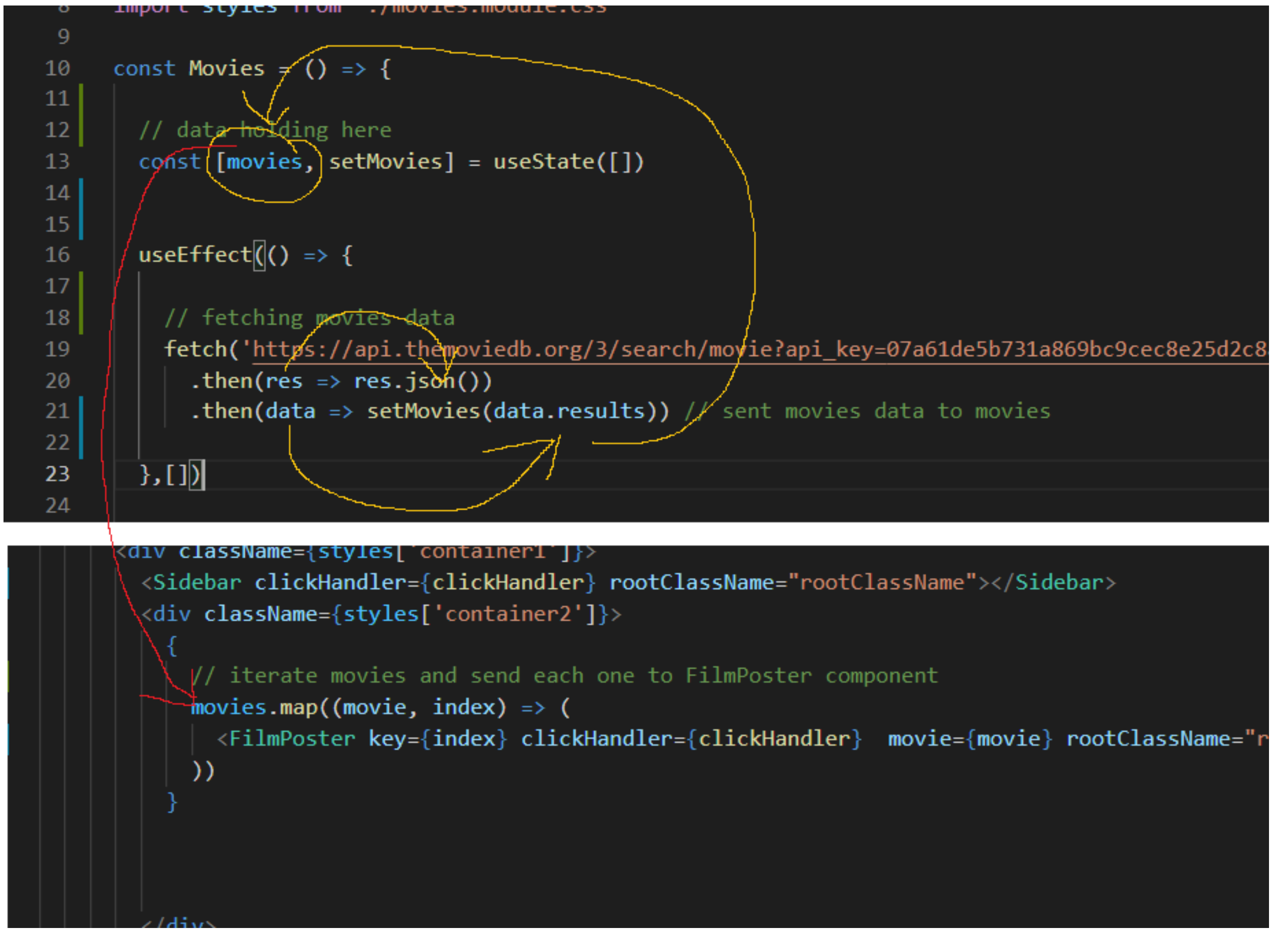
```
useEffect(() => {

  // fetching movies data
  fetch('https://api.themoviedb.org/3/search/movie?api_key=07a61de5b731a869bc9cec8e25d2c8a8&query=batman')
    .then(res => res.json())
    .then(data => setMovies(data.results)) // sent movies data to movies

},[])
```

Figure 9: use of the fetch command.

With “.then(res=>res.json())” in the first chain, the data is fetched and converted into json readable data and with “.then(data=> setMovies(data.results))” in the second chain, the now readable data is sent to a stat variable (movies) by the setMovies method. Finally, we were able to use the json-data in the return scope to print the movies-property values like ‘original_title’, ‘poster_path’ ‘time’,… (Fig. 10).



```
const Movies = () => {

  // data holding here
  const [movies, setMovies] = useState([])


  useEffect(() => {

    // fetching movies data
    fetch('https://api.themoviedb.org/3/search/movie?api_key=07a61de5b731a869bc9cec8e25d2c8
      .then(res => res.json())
      .then(data => setMovies(data.results)) // sent movies data to movies

  },[])

```

```
<div className={styles['container1']}>
  <Sidebar clickHandler={clickHandler} rootClassName="rootClassName"></Sidebar>
  <div className={styles['container2']}>
    {
      // iterate movies and send each one to FilmPoster component
      movies.map((movie, index) => (
        <FilmPoster key={index} clickHandler={clickHandler}  movie={movie} rootClassName="r
      ))
    }



  </div>
```

Figure 10: mapping each movie from API to a component in website.

2.4 GitHub Actions and launch via Heroku (stage 3)

2.4.1 *What is GitHub*

At a high level, GitHub is a website and cloud-based service that helps developers store and manage their code, as well as track and control changes to their code. To understand exactly what GitHub is, you need to know two connected principles:

- Version control
- Git

Version control helps developers track and manage changes to a software project's code. As a software project grows, version control becomes essential.

Specifically, Git is a distributed version control system, which means that the entire codebase and history is available on every developer's computer, which allows for easy branching and merging.

2.4.2 *Use of GitHub*

During this project we made use of following command sequence:

> git clone https://github.com/bloatedgoat/Trends-II-MovieDatabase-React (First time)

Suppose developer needs to create one specific repository of GITHUB in their local copy from the specific remote location. Then they have to execute clone command for copying the same remote repository in the local environment in specific location.

> git pull *Helping for updating all the newest commit in the mapping local repository.*

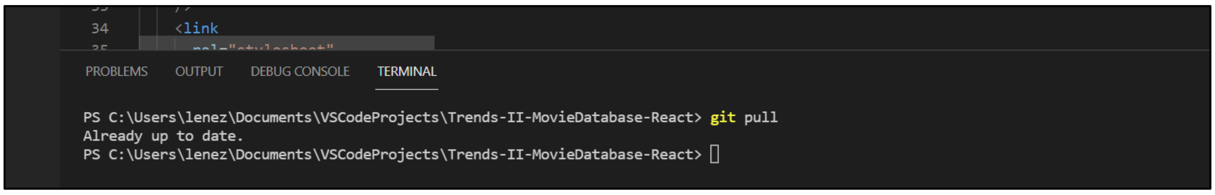
> git add . *Adding changes from current directory.*

> git commit -m "First git commit of static movie page from node.js"

Commit all the required changes.

> git push -u origin main

Pushing entire local repository branch to the remote repository, extra utility of using this command is also remembered for this specific branch for future reference.



```
PS C:\Users\lenez\Documents\VSCodeProjects\Trends-II-MovieDatabase-React> git pull
Already up to date.
PS C:\Users\lenez\Documents\VSCodeProjects\Trends-II-MovieDatabase-React> 
```

Use of terminal command

2.4.3 *What is GitHub action?*

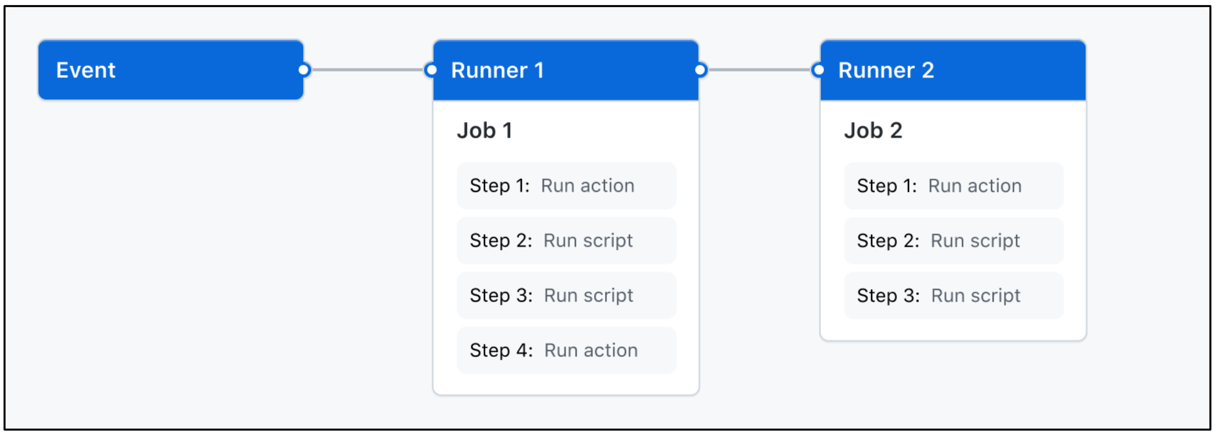
First launched in 2018 as a platform-native automation tool, GitHub Actions has evolved to give developers powerful automation and CI/CD (continuous integration/continuous deployment) capabilities right next to your code in GitHub.

At its core, GitHub Actions is designed to help simplify workflows with flexible automation and offers easy-to-use CI/CD capabilities built by developers for developers.

Compared with other automation or CI/CD tools, GitHub Actions offers native capabilities right in your GitHub flow. It also makes it easy to leverage any of the 10,000+ pre-written and tested automations and CI/CD actions in the GitHub Marketplace as well as the ability to write your own with easy-to-use YAML files.

The best part? GitHub Actions responds to webhook events. That means you can automate any workflow based on a webhook trigger in your GitHub repository—whether it's from an event on GitHub or from a third-party tool.

You can configure a GitHub Actions workflow to be triggered when an event occurs in your repository, such as a pull request being opened or an issue being created. Your workflow contains one or more jobs which can run in sequential order or in parallel. Each job will run inside its own virtual machine runner, or inside a container, and has one or more steps that either run a script that you define or run an action, which is a reusable extension that can simplify your workflow.



Example of a workflow process

A workflow is a configurable automated process that will run one or more jobs. Workflows are defined by a YAML file checked in to your repository and will run when triggered by an event in your repository, or they can be triggered manually, or at a defined schedule.

An event is a specific activity in a repository that triggers a workflow run.

A job is a set of steps in a workflow that execute on the same runner. Each step is either a shell script that will be executed, or an action that will be run. Steps are executed in order and are dependent on each other. Since each step is executed on the same runner, you can share data from one step to another.

An action is a custom application for the GitHub Actions platform that performs a complex but frequently repeated task. Use an action to help reduce the amount of repetitive code that you write in your workflow files.

A runner is a server that runs your workflows when they're triggered. Each runner can run a single job at a time. GitHub provides Ubuntu Linux, Microsoft Windows, and macOS runners to run your workflows; each workflow run executes in a fresh, newly-provisioned virtual machine.

At a certain point we had a problem with our GitHub (accidental push and we could not reverse it back for some reason). Here we have created a branch called newBranch.

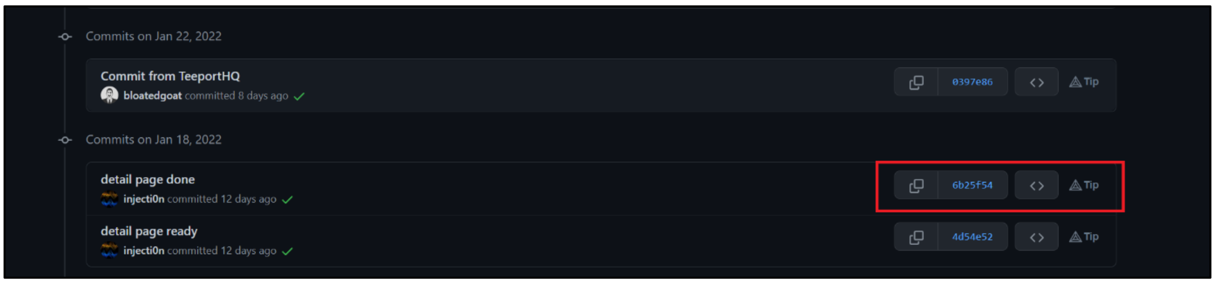
We took the following steps:

In terminal we entered following commands:

- `git clone https://github.com/bloatedgoat/Trends-II-MovieDatabase-React.git`

  Started a new clone for less problems.
- Go to current directory. (`cd .\Trends-II-MovieDatabase-React\`)
- `git checkout -b newBranch 6b25f54f7fecbc41943cc621cb9219dbddba49c4`

  Here we created a new local branch. The hash is from the last good version that we pushed:



Commits on Jan 22, 2022

Commit from TeeportHQ
bloatedgoat committed 8 days ago ✓ — 0397e86

Commits on Jan 18, 2022

detail page done
injecti0n committed 12 days ago ✓ — 6b25f54

detail page ready
injecti0n committed 12 days ago ✓ — 4d54e52

- `git push origin newBranch`

  Here we push out the branch to GitHub.
- `git commit -m "This is the old version to update Heroku page from newBranch"`

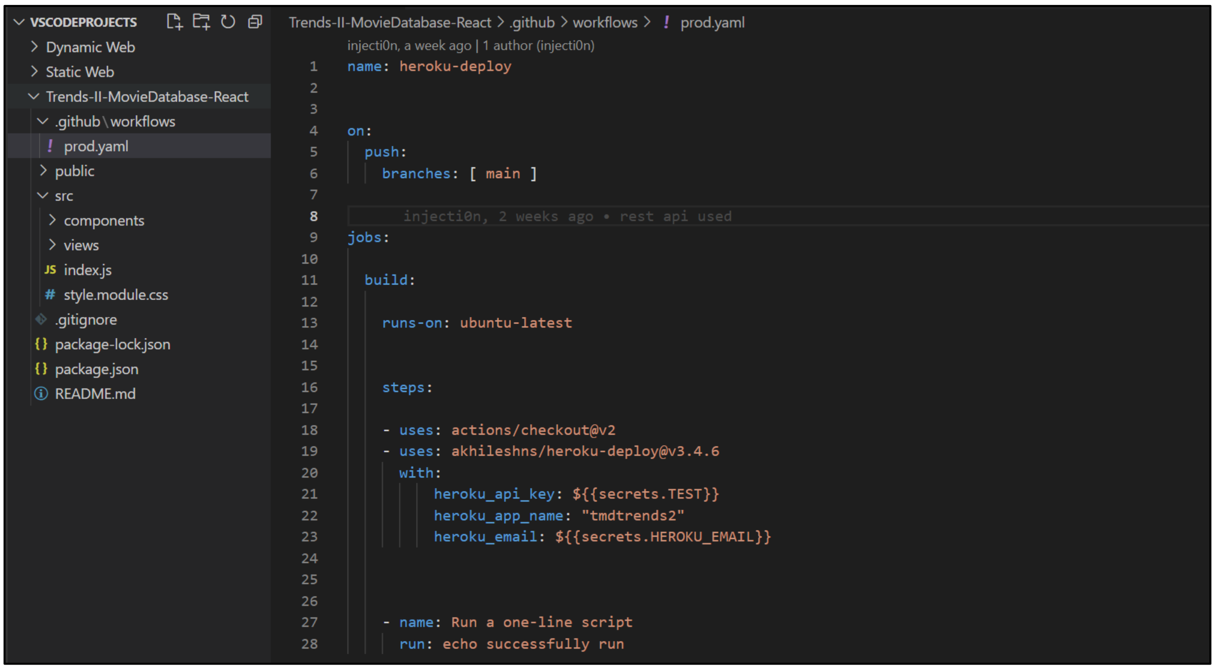
  We made a simple change and committed the change with some explanation.
- `git pull origin newBranch`

  You have to pull before you do a push to check for changes.
- `git push -u origin newBranch`

  Old project is pushed to new branch called newBranch.

2.4.4 Use of GitHub actions

GitHub Actions uses YAML syntax to define the workflow. Each workflow is stored as a separate YAML file in your code repository, in a directory called “. github/workflows”.



```
VSCODEPROJECTS
> Dynamic Web
> Static Web
v Trends-II-MovieDatabase-React
  v .github\workflows
    ! prod.yaml
  > public
  v src
    > components
    > views
    JS index.js
    # style.module.css
  .gitignore
  {} package-lock.json
  {} package.json
  ⓘ README.md
```

```
Trends-II-MovieDatabase-React > .github > workflows > ! prod.yaml
injecti0n, a week ago | 1 author (injecti0n)
name: heroku-deploy


on:
  push:
    branches: [ main ]

    injecti0n, 2 weeks ago • rest api used
jobs:

  build:

    runs-on: ubuntu-latest


    steps:

    - uses: actions/checkout@v2
    - uses: akhileshns/heroku-deploy@v3.4.6
      with:
        heroku_api_key: ${{secrets.TEST}}
        heroku_app_name: "tmdtrends2"
        heroku_email: ${{secrets.HEROKU_EMAIL}}



    - name: Run a one-line script
      run: echo successfully run
```

Example of the used .yaml file in our project.

Steps Taken:

1. In your repository, create the **.github/workflows/** directory to store your workflow files.
2. In the .github/workflows/ directory, create a new file called **prod.yml** and add the following code:

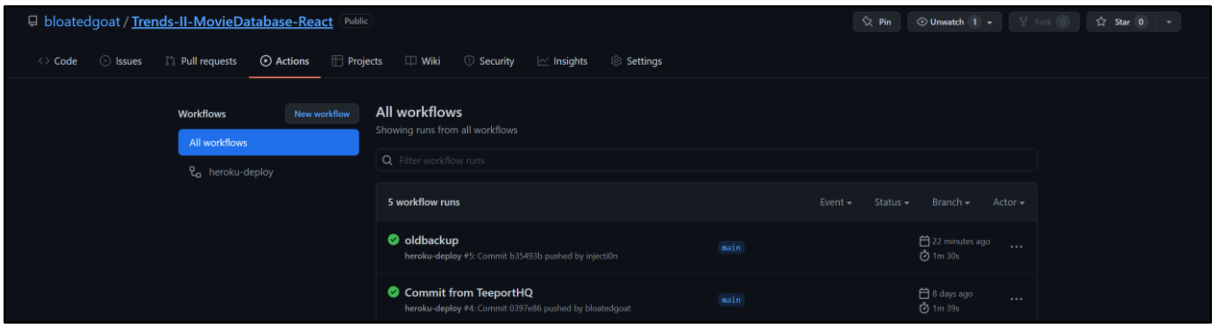
```
name: heroku-deploy
on:
  push:
    branches: [ main ]                //Can also use newBranch
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - uses: akhileshns/heroku-deploy@v3.4.6
      with:
          heroku_api_key: ${{secrets.TEST}}
          heroku_app_name: "tmdtrends2"
          heroku_email: ${{secrets.HEROKU_EMAIL}}
    - name: Run a one-line script
      run: echo successfully run
```

3. Commit these changes and push them to your GitHub repository.

Your new GitHub Actions workflow file is now installed in your repository and will automatically run each time someone pushes a change to the repository.

Now go to your Heroku account and go to Account Settings. Scroll to the bottom until you see API Key. Copy this key and go to your project's repository on GitHub. (See chapter 2.4.4)

In your Repo, go to Settings -> Secrets and click on "New Secret". Then enter HEROKU_API_KEY as the name and paste the copied API Key as the value.
You can now push your project to GitHub and it will be automatically deployed to Heroku henceforth.
Every time we push a change to the GitHub repository, our heroku server gets updated and we can see the results immediately on https://tmdtrends2.herokuapp.com/ .



All workflows
Showing runs from all workflows

5 workflow runs

oldbackup
heroku-deploy #5: Commit b35493b pushed by injecti0n
main
22 minutes ago
1m 30s

Commit from TeeportHQ
heroku-deploy #4: Commit 0397e86 pushed by bloatedgoat
main
8 days ago
1m 39s

History of all triggered workflows by GitHub actions

2.4.5 What is Heroku

"Heroku is a cloud platform that lets companies build, deliver, monitor and scale apps — we're the fastest way to go from idea to URL, bypassing all those infrastructure headaches."

To understand what Heroku is, we're going to start with what it's not. You may have realized that Heroku runs on Amazon Web Services (AWS), and now you're asking yourself why you don't deploy to AWS and bypass Heroku entirely. First of all, Heroku and AWS are not the same things.

AWS is an Infrastructure as a Service (IaaS) provider, meaning they are responsible for managing large, shared data centers. These data centers are what we call "the cloud". Companies like AWS, Azure, and Google have all created IaaS so that developers can pay to host their applications in these data centers instead of building servers themselves.

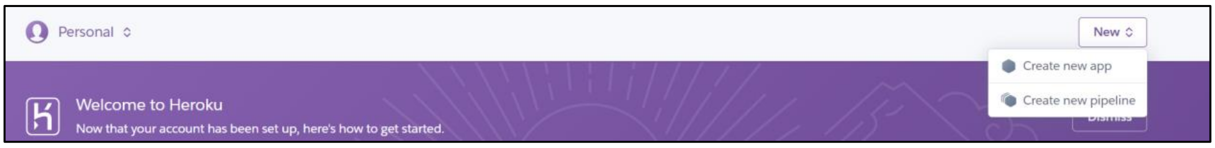
This is a great trade-off but due to the nature of their business, IaaS providers are more concerned with running the data centers than the developer's experience working with them. This means there is a high level of knowledge of AWS is required to keep your apps running, especially at scale.

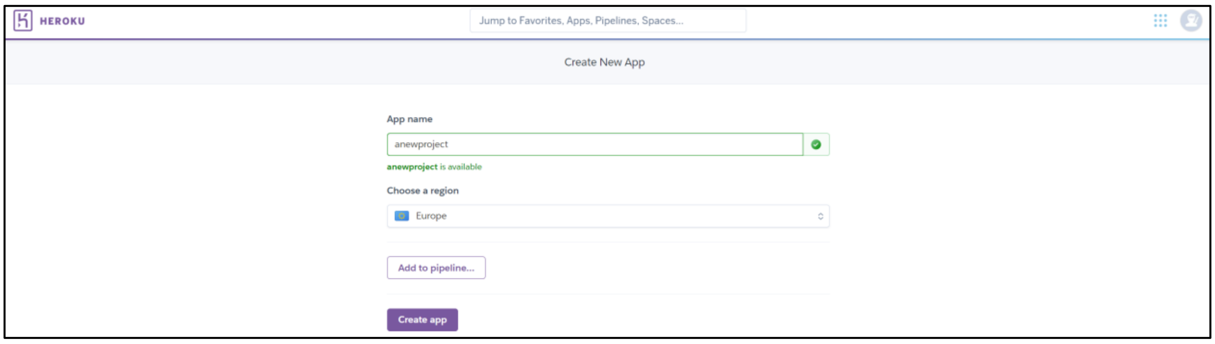
Heroku, on the other hand, is a Platform as a Service that sits on top of AWS to provide an experience that is specifically designed to make developers lives easier. For example, in order to keep an application running at scale on Heroku, it only takes knowledge of a few commands on the Heroku CLI and Dashboard. These commands can easily be found in Heroku's documentation.

2.4.6 Setting up Heroku

Create a Heroku app and save the app's name and email associated with your account. Then go to Account settings and copy your API Key.

Creating a new Heroku application:


Personal
New
Create new app
Create new pipeline
Welcome to Heroku
Now that your account has been set up, here's how to get started.

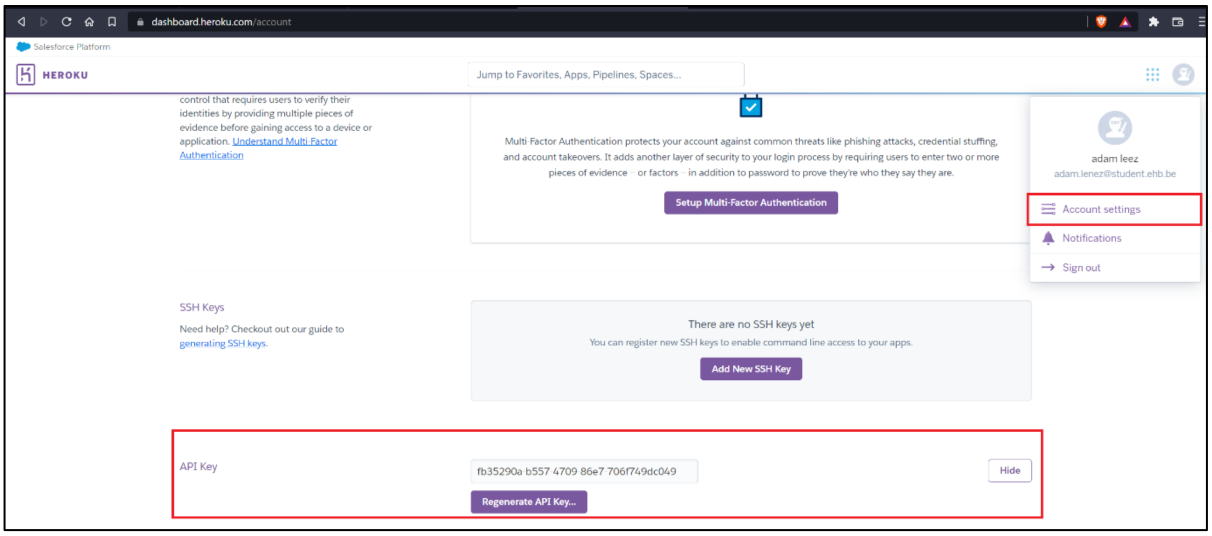


App name: anewproject

anewproject is available

Choose a region: Europe

Go to settings and copy the API key:


dashboard.heroku.com/account

control that requires users to verify their identities by providing multiple pieces of evidence before gaining access to a device or application. Understand Multi Factor Authentication

Multi Factor Authentication protects your account against common threats like phishing attacks, credential stuffing, and account takeovers. It adds another layer of security to your login process by requiring users to enter two or more pieces of evidence – or factors – in addition to password to prove they're who they say they are.

SSH Keys

Need help? Checkout out our guide to generating SSH keys.

There are no SSH keys yet

You can register new SSH keys to enable command line access to your apps.

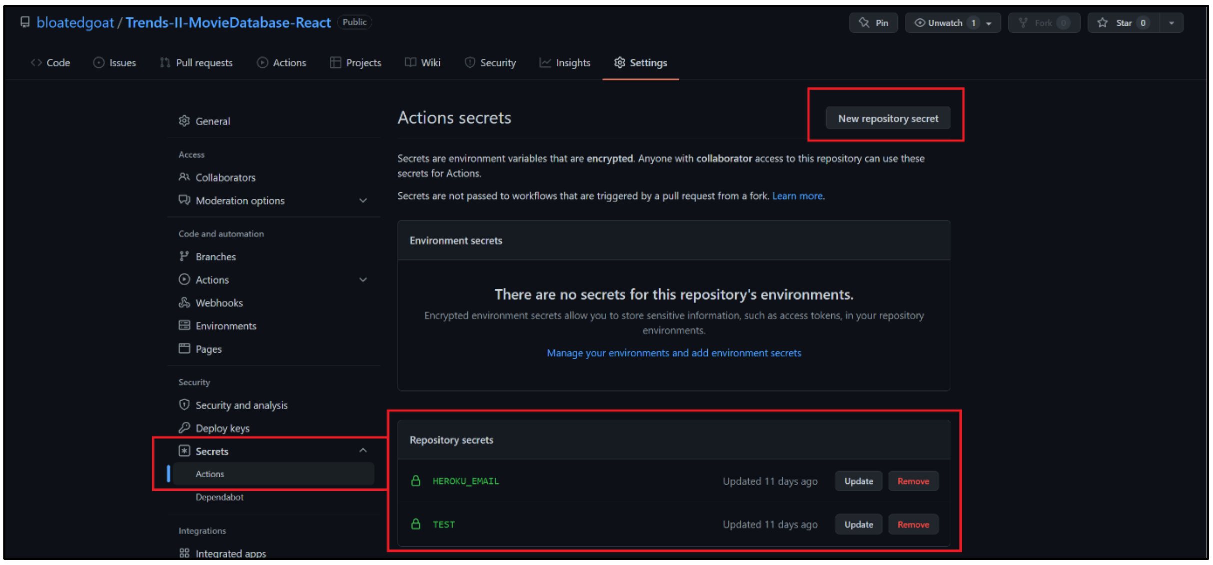
API Key: fb35290a b557 4709 86e7 706f749dc049

adam leez
adam.lenez@student.ehb.be

To create a secure integration between systems and linking accounts safely, we need to protect our passwords, secret keys and tokens to authenticate themselves. That's where the **GitHub Actions secret** comes in. It provides a secure vault to store your confidential information, and a simple mechanism to access those secret tokens in your GitHub Actions scripts.

Go to your GitHub repository's Settings > Secrets and set the secrets:

- HEROKU_API_KEY
- (HEROKU_APP_NAME)
- HEROKU_EMAIL


bloatedgoat / Trends-II-MovieDatabase-React

Actions secrets

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

Environment secrets

There are no secrets for this repository's environments.

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

Manage your environments and add environment secrets

Repository secrets

HEROKU_EMAIL	Updated 11 days ago	Update	Remove
TEST	Updated 11 days ago	Update	Remove

Where to find the API keys.

3. Conclusion

General

Low code development offers a way to build business applications quickly, cheap and with the resources you already have.

However, this method is not suited for developing complex enterprise level applications.

This way of developing applications will evolve and eventually even complex applications will become possible. There still we be a deficit of professional developers, so this is not the solution to all problems businesses are currently facing.

PoC

While working on a PoC we noted that indeed, low code platforms can kickstart the user and get you going quickly. However, the quality of the auto generated code is not optimal, meaning it has to be tweaked afterwards in order to get the desired result.

You are also limited in what you can do in this kind of development environment. You get a set of objects you can drag and drop onto your canvas and that's what you've got to work with.

For our PoC we therefore believe it would have been quicker and easier if we coded the static website ourselves. However, we also must consider that teleportHQ is an open-source platform and that it is possibly more limited than a licensed platform. Maybe with a licensed platform we would have more functionalities and possibilities and we would have to adjust the code afterwards.

To conclude: for our PoC the low-code platform teleportHQ didn't offer an advantage in developing the website, but we believe that for more static websites that are less complex the platform will prove to be advantageous.