Report on IDE differences

Abstract

In the tech environment, where the process of writing code is involved, having a good IDE is a priority. Just by writing the code in a simple text editor is not enough, because every developer is looking for accessibility and ease of writing code. Depending on the performance or type of features that an IDE provide, the developer can choose from a large variety of software for writing code.

Types of IDEs

On the market there is a large variety of IDEs, and choosing the perfect one can only be done, by working a certain amount of time with that specific code editor. The list of IDEs is long, but we are going to talk about the most used.

The first IDE tested was Atom, which is a software developed by GitHub. Even though Atom was developed relatively recent, compared to other IDEs on the market, it's power was brought into the attention of the developers, so it gained a huge recognition in a short amount of time.
One important aspect of the software is the ease of customization, and package installer. So one thing that Atom is good at, is the ease of theming the coding editor.

On the other side, one huge downside of Atom is the performance. When we are dealing with a huge amount of files, the software show a constant delay between switching the files. We tested this with a large number of files from the node_modules, that contains all the installed packages. When time is an important aspect of the development process, this issue can generate many problems. The analysis was made in 2019, as the project states, but after we investigated even further, it looks like the speed was the main major issue one year ago. Since then it becomes less of a problem, but there is still a issue.

The second IDE tested was VS Code, which is a software developed by Microsoft. The same as with Atom, VS Code is a relatively new IDE, being release in 2015, but the power and the way it works brought it into the attention of developers all over the world. What we liked about this code editor, was the plugin ecosystem. The extension is built in and it is really easy to install anything that is necessary for the IDE.

Speaking about the way the software works, VS Code is a lot powerful compared to Atom, because there is no real lag in the development process, even with a large number of files. For us as developers, one important aspect is the debugging process, and one thing VS Code is good at is good at is debugging. Another thing that we noticed during the testing process, was the fact that this software contains a built-in feature that communicates directly to GitHub. So instead of using the regular commands in the terminal, we could push the code directly from the code editor.

Choosing an IDE

We've decided to use IntelliJ for the exercise. IntelliJ is a fully fledged IDE that contains all the tools a developer might think of and want. It is language agnostic and highly customizable for the languages, frameworks and data sources that the developer is working with.

We are choosing Intellij because we feel it's a solid IDE and the right choice for the exercise compared the other options which are closer to being text editors. But with Atom and VSCode the lines have become quite blurred.

Some of us in the group have never used an IDE before so that's also why we wanted to try it out. But because of the limited scope of the exercise it's hard to dive deep into all the features. Reflecting on this it's obvious that we might've ended up using IntelliJ exactly like we would a text editor. Getting to know your IDE is a long process of uncovering new features often under serendipitous circumstances.

Experiences while writing the code

When you write a more complex piece of code, in order to create a better view and hierarchy, it was important to have a good code editor, that highlights the code. Even though, writing the algorithms require more of a logic thinking than a visual thinking, and it can be done on a whiteboard as qualitative as on a IDE, the software helped us to have a better view of the indentation of the code. With a code editor that doesn't create a good visual aspect of the code, we can fall in writing slow algorithms, jumping from a linear time complexity O(n) to a quadratic time complexity O(n2), which is the worst we can get while trying to write the algorithms.

IntelliJ is the first IDE to bring Intellisense to the world but we never got to take it for a ride because we had 2 files at most during each execution. We also never got to explore the rich file navigation / searching system.

Conclusion

For the purpose of this exercise, we only got to scratch the surface of the IDEs we tested, because they are a lot bigger than we got to test. But from what we analyzed choosing the best software it is base on the type of the developer. We choose IntelliJ because it includes many of the characteristics of a good coding editor.

Appendix I - MergeSort

```
function mergeSort (arrDdatas) {
   if (arrDdatas.length === 1) {
     return arrDdatas
   }
    const middleVal = Math.floor(arrDdatas.length / 2)
   const leftVal = arrDdatas.slice(0, middleVal)
   const rightVal = arrDdatas.slice(middleVal)
    return merge(
     mergeSort(leftVal),
     mergeSort(rightVal)
   )
 }
  function merge (left, right) {
   let result = []
   let indexLeft = 0
   let indexRight = 0
    while (indexLeft < left.length && indexRight < right.length) {
     if (left[indexLeft] < right[indexRight]) {
       result.push(left[indexLeft])
       indexLeft++
     } else {
       result.push(right[indexRight])
       indexRight++
     }
   }
    return result.concat(left.slice(indexLeft)).concat(right.slice(indexRight))
 }
 const list = [5, 2, 1, 3, 9, 12, 21, 8, 5, 9]
 console.log(mergeSort(list))
```

Appendix I - InsertionSort

```
function insertionSort (arrItems) {
  for (var i = 0; i < arrItems.length; i++) {
    let value = arrItems[i]
    for (var j = i - 1; j > -1 && arrItems[j] > value; j--) {
      arrItems[j + 1] = arrItems[j]
    }
    arrItems[j + 1] = value
  }

  return arrList
}
 const arrList = [54, 26, 93, 17, 77, 31, 44, 55, 20]
console.log(insertionSort(arrList))
```