

E-Nano2020 README

Esta es la documentación del proyecto del 2020 de nombre de E-Nano2020 el cual es una aplicación web a la que se le ingresa código en formato E-Nano el cual es transpilado a Java para luego ser compilado; luego usando el nombre del archivo se puede ejecutar el `main` de dicho código.

Se recomienda al lector leer la documentación completamente antes de ejecutar el programa.

Pre-requisitos

Para poder ejecutar este proyecto se requiere tener lo siguiente instalado en el computador (preferiblemente la última versión):

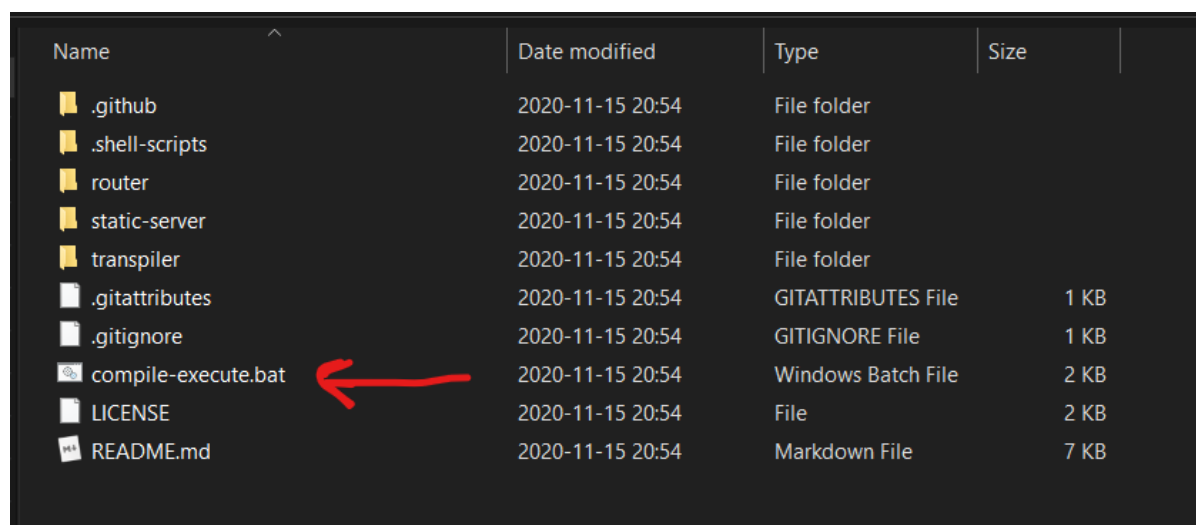
1. **Java 14**
2. **Gradle**
3. **SWI-Prolog**
4. Un navegador, preferiblemente **Google Chrome**.

Ejecución del proyecto

Compilación y ejecución de los servidores

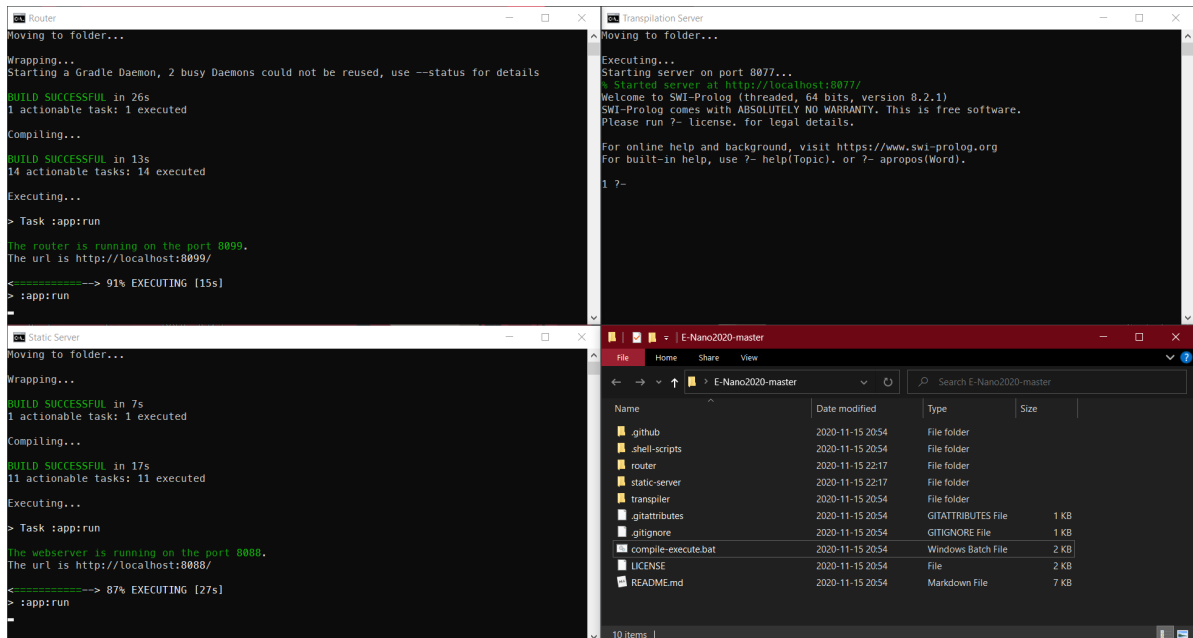
Para ejecutar el proyecto se debe ejecutar el archivo `compile-execute.bat` el cual se encuentra en la carpeta raíz del proyecto a como se puede ver a continuación. Este archivo ejecuta otros tres scripts distintos en diferentes ventanas los cuales compilan y ejecutan cada uno de los servidores del proyecto, los cuales son:

- Servidor de router para compilar código Java y extraer información de la base de datos.
- Servidor web estático que se encarga de servir los archivos estáticos.
- Servidor de Prolog el cual se encarga de transpilar el código E-Nano a Prolog.



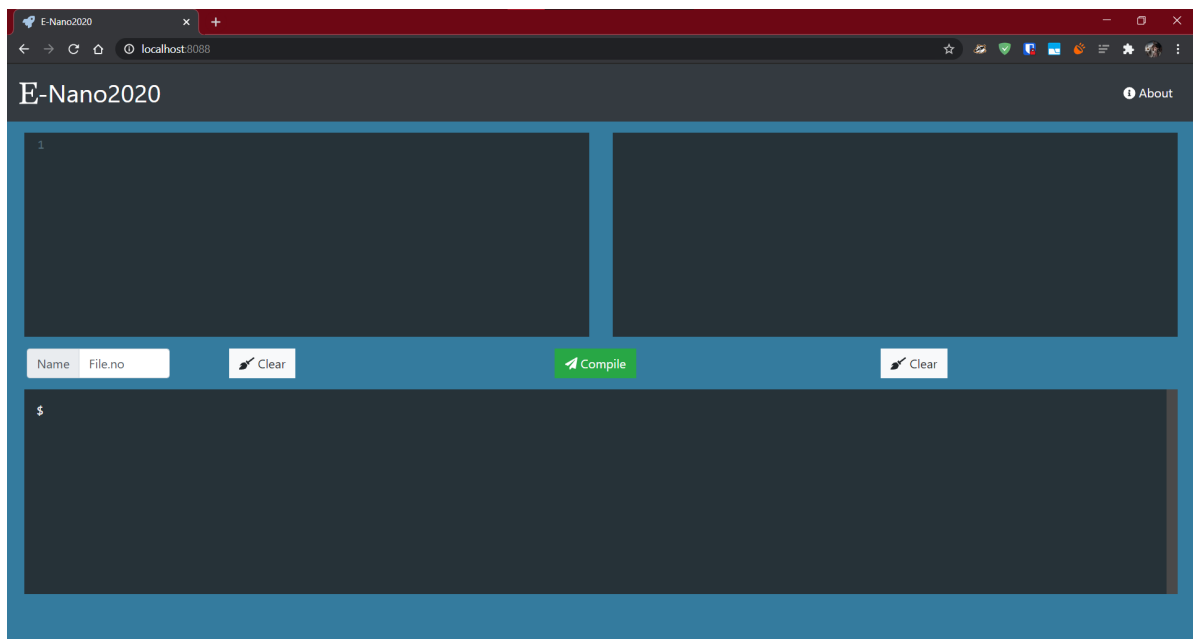
Name	Date modified	Type	Size
.github	2020-11-15 20:54	File folder	
.shell-scripts	2020-11-15 20:54	File folder	
router	2020-11-15 20:54	File folder	
static-server	2020-11-15 20:54	File folder	
transpiler	2020-11-15 20:54	File folder	
.gitattributes	2020-11-15 20:54	GITATTRIBUTES File	1 KB
.gitignore	2020-11-15 20:54	GITIGNORE File	1 KB
compile-execute.bat	2020-11-15 20:54	Windows Batch File	2 KB
LICENSE	2020-11-15 20:54	File	2 KB
README.md	2020-11-15 20:54	Markdown File	7 KB

Una vez los tres servidores se compilaron y ejecutaron satisfactoriamente, a como se puede ver en la siguiente imagen se puede proceder a ejecutar la aplicación web en el navegador ingresando la url del servidor estático, la cual por defecto es <http://localhost:8088/>, en el navegador.



Ingreso a la aplicación web

Una vez se ingresó la url del servidor estático en el navegador se va a ver una pantalla como la siguiente:



Panel de código

En el panel que está arriba a la derecha se puede ingresar código E-Nano el cual al apretar el botón que dice **Compile** se va a transpilar a código de Java para luego ser compilado y guardado en el servidor **Router**. Los errores o mensajes de compilación o transpilación se pueden ver en el panel de mensajes.

Este panel se puede limpiar con el botón que está justo debajo de este que dice **Clear**.

Adicionalmente, el código de Java transpilado se puede ver en la consola del servidor **Router** una vez que se envió la petición de compilación a este servidor.

Panel de mensajes

En este panel, el cual se encuentra arriba a la izquierda, se pueden ver mensajes de éxito o error de compilación y/o transpilación una vez que algún código fue compilado. Los mensajes se muestran en código de colores según su tipo:

- Los mensajes **rojos** son de **error**, significan que la compilación/transpilación no se pudo llevar con éxito.
- Los mensajes **verdes** son de **éxito**, significan que la compilación se llevo a cabo con éxito.
- Los mensajes **amarillos** son de **advertencia**, sirven para llamar la atención del usuario sobre algún detalle peculiar que deben tener presente.
- Los mensajes **blancos** son de **anotación**, significan algo que el usuario puede querer saber pero no necesariamente algo que deban arreglar.

Campo de nombre del archivo

Este campo se puede ver justo **debajo** del **panel de código** y sirve para ingresar el nombre que se desea que el archivo tenga al ser compilado y transpilado, esto afecta el nombre de la clase durante la compilación y el identificador con el que se va a guardar el archivo ejecutable en el servidor.

Un nombre válido que se puede ingresar en este campo es un nombre formado por una letra inicial y luego letras o números finalizando con un `.no`. Por ejemplo: `File.no`.

Si este campo tiene un dato erróneo se va a mostrar con un borde rojo y el botón de compilar se va a desactivar.

Si se compila un archivo cuyo nombre fue utilizado antes para otro archivo se va a mostrar una advertencia en el panel de mensajes.

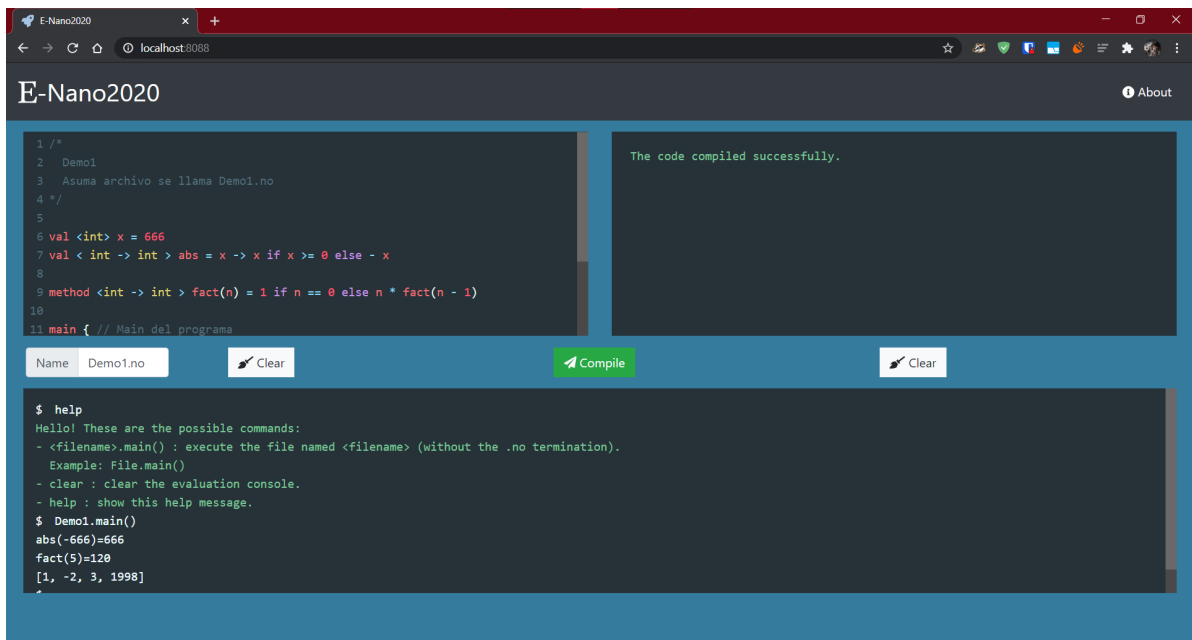
Panel de ejecución

Este panel se puede abajo del todo, debajo de los dos paneles de mensajes y código.

En este panel se puede ingresar algunos de los siguientes comandos, y una vez que se presiona la tecla `enter` este comando se va a ejecutar, los comandos son los siguientes:

- `<filename>.main()`: este comando sirve para ejecutar el archivo llamado `<filename>` el cual no debe tener la terminación `.no` y debió ser compilado con anterioridad. Un ejemplo es: `File.main()`.
- `clear`: sirve para limpiar el panel de ejecución.
- `help`: sirve para mostrar un mensaje de ayuda para saber los comandos que se pueden utilizar.

Ejemplo de uso del programa



Configuración

Algunos parámetros de los servidores se pueden configurar, esto se hace modificando el archivo `server.properties` dentro de `/resources` del servidor (e.g. `/router/resources/config/server.properties` o `/static-server/resources/config/server.properties`).

Port

Este parámetro está disponible para el servidor de router y el servidor estático.

Ejemplo:

```
port=8080
```

Esto haría que el servidor utilice el puerto `8080`.

Estructura del proyecto

La estructura del proyecto es la siguiente:

```
\ [root]
├──.github
│   └──workflows
├──.shell-scripts
├──router
│   ├──app
│   │   └──src
│   │       ├──main
│   │       │   ├──java
│   │       │   │   ├──org
│   │       │   │   │   ├──una
│   │       │   │   │       └──app
│   │       └──test
│   │           ├──java
│   │           │   ├──org
│   │           │       └──una
```

```
├── app
├── buildSrc
│   ├── src
│   │   ├── main
│   │   └── groovy
├── compiler
│   ├── src
│   │   ├── main
│   │   │   ├── java
│   │   │   │   ├── org
│   │   │   │   │   ├── una
│   │   │   │   │   └── compiler
├── gradle
│   └── wrapper
├── resources
│   └── config
├── utilities
│   ├── src
│   │   ├── main
│   │   │   ├── java
│   │   │   │   ├── org
│   │   │   │   │   ├── una
│   │   │   │   │   └── utilities
├── static-server
│   ├── app
│   │   ├── src
│   │   │   ├── main
│   │   │   │   ├── java
│   │   │   │   │   ├── org
│   │   │   │   │   │   ├── una
│   │   │   │   │   │   └── app
│   │   │   └── test
│   │   │       ├── java
│   │   │       │   ├── org
│   │   │       │   │   ├── una
│   │   │       │   │   └── app
│   ├── buildSrc
│   │   ├── src
│   │   │   ├── main
│   │   │   └── groovy
│   ├── gradle
│   │   └── wrapper
│   ├── resources
│   │   ├── config
│   │   └── web
│   │       ├── dist
│   │       │   ├── css
│   │       │   └── js
│   │       ├── public
│   │       └── src
│   │           ├── components
│   │           └── plugins
│   └── utilities
│       ├── src
│       │   ├── main
│       │   │   ├── java
│       │   │   │   ├── org
│       │   │   │   └── una
```

└─transpiler

└─utilities

Los archivos web están en la carpeta `/static-server/resources/web/dist`; el resto del proyecto se maneja de la forma estándar de Java.

Extras

MongoDB

De extras está implementado la extra de traer los datos de **About** de una base de datos *MongoDB*.

A continuación se muestra la estructura de las colecciones en *MongoDB*.

The screenshot displays the MongoDB Atlas web interface. On the left, a sidebar shows the database structure with a tree view containing 'ENANO2020' and its sub-collections 'INFO' and 'TEAM'. The main panel is divided into two sections. The top section, titled 'ENANO2020.INFO', shows a collection with 1 document and a total index size of 20KB. It displays a single document with fields like '_id', 'project', 'course', 'instance', 'cycle', 'organization', 'projectSite', and 'code'. The bottom section, titled 'ENANO2020.TEAM', shows a collection with 5 documents and a total index size of 36KB. It displays three documents, each with fields like '_id', 'firstName', 'surNames', and 'id'. The interface includes search filters, 'Find' and 'Reset' buttons, and an 'INSERT DOCUMENT' button.

ENANO2020.INFO

COLLECTION SIZE: 197B TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 20KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER {"filter": "example"}

Find Reset

QUERY RESULTS 1-1 OF 1

```
{
  "_id": ObjectId("5f7b387df082e02fb79e2372"),
  "project": "Enano",
  "course": "EIF408",
  "instance": "HRC50858",
  "cycle": "II-2020",
  "organization": "UNA",
  "projectSite": "https://github.com/datwaft/E-Nano2020",
  "code": "03-10am"
}
```

ENANO2020.TEAM

COLLECTION SIZE: 456B TOTAL DOCUMENTS: 5 INDEXES TOTAL SIZE: 36KB

Find Indexes Schema Anti-Patterns Aggregation Search Indexes

INSERT DOCUMENT

FILTER {"filter": "example"}

Find Reset

QUERY RESULTS 1-5 OF 5

```
{
  "_id": ObjectId("5f7ba2792936a87fbbcdc25e"),
  "firstName": "David",
  "surNames": "Guevara Sánchez",
  "id": "402450355"
}
```

```
{
  "_id": ObjectId("5f7ba29a2936a87fbbcdc25f"),
  "firstName": "Joy",
  "surNames": "Bonilla Flej",
  "id": "402360421"
}
```

```
{
  "_id": ObjectId("5f7ba2a62936a87fbbcdc260"),
  "firstName": "Jose",
  "surNames": "Barrantes Araya",
  "id": "207600954"
}
```

Typescript

Otro extra que está implementado es el de utilizar *Typescript* para hacer el servidor web, en este caso fue implementado usando *VueCLI* con *Typescript*.

Créditos

Grupo de trabajo No.03:

- David Alberto Guevara Sánchez\
402450355
- Joy Bonilla Fley\
402360421
- José Barrantes Araya\
207600954
- Natalia Solano Azofeifa\
117290958
- Luis David Villalobos González\
117540697

Horario: 10am.