

Reimaginando la revisión de código con RAG para salvarnos de LGTM

<https://www.watermelontools.com/post/reimagining-code-review-with-rag-to-save-us-from-lgtm>

El ciclo de vida de una aplicación de revisión de código impulsada por RAG

Los desarrolladores dedican el 30% de su tiempo a revisiones de código. Aunque lleva mucho tiempo, es un paso crucial. Después de todo, arreglar errores en un entorno de producción puede ser costoso.

-

Aquí en esta empresa, estamos abordando este problema de frente, aprovechando la era actual de LLMs para acelerar las revisiones de código con un enfoque impulsado por IA.

-

En este post, daremos a conocer el funcionamiento interno de la aplicación GitHub de [Watermelon](#) impulsada por RAG para la revisión de código, y cómo queremos que las relaciones públicas tengan más contexto que un comentario LGTM.

-

Una rápida introducción a la RAG

En el ámbito de la ingeniería de IA, Retrieval Augmented Generation (RAG) se refiere a la extracción de información de fuentes externas para mejorar los resultados de los LLM.

-

Tomen, por ejemplo, ChatGPT, que ahora utiliza RAG. Con [la llamada](#) de [la función](#), puedes pedir que ["nunca la web y trae la temperatura actual de San Francisco"](#). Recupera la información de Bing y le presenta el hecho, eludiendo la limitación habitual de no tener datos actuales.

-

En nuestro caso estamos haciendo RAG no navegando por la web con la llamada de función, sino construyendo integraciones OAuth a diferentes fuentes de información de la empresa que nos permiten recuperar información llamando a diferentes API. Estamos usando RAG para proporcionar un contexto más rico para tirar de solicitudes, yendo más allá [del "LGTM" demasiado común](#). Vamos a sumémonos en algunos hechos interesantes de la LGTM.

Sólo LGTM

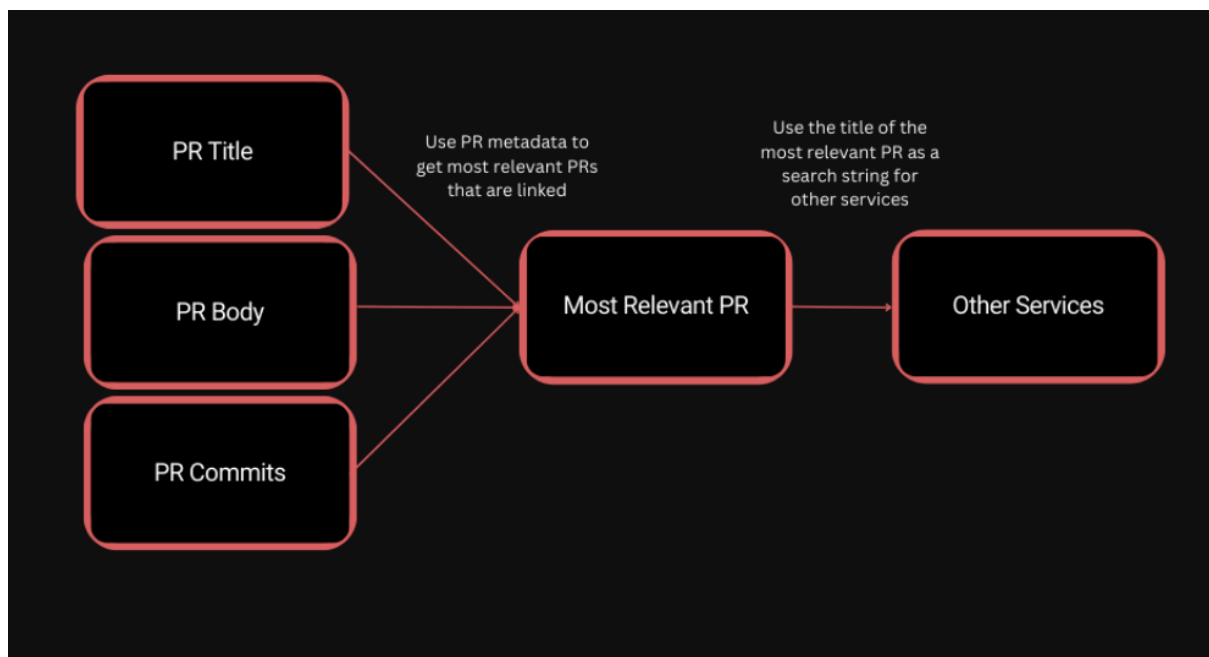
- 65% de las relaciones públicas contienen un mensaje de aprobación que sólo tiene un lgtm
- El 40% de las PR no contienen una descripción
- Las partes interesadas no técnicas aportan una visión holística del proceso de revisión del código, pero el 50% de ellas afirman que la revisión del código es un proceso muy técnico y complejo. Esto es muy importante. Muchas veces los cambios se revieren no porque hay algo malo a nivel técnico, sino porque un cambio no cumplieron con los requisitos de negocio

Introducción a la búsqueda de contexto de código

La primera acción que tomamos hacia la contextualización de relaciones públicas más allá de lgtm es rastrear el contexto de código vinculado a una nueva PR.

En nuestro caso, el primer paso en nuestro proceso de RAG es recuperar un nuevo metadatos de PR. Este no es sólo el primer paso en el proceso, sino el que más influye en cómo se comportará el flujo de la recuperación del resto de la recuperación. Para una relaciones públicas recuperamos:

- El título de PR.: Que más tarde usamos como parámetro de búsqueda para golpear otras API que un equipo optativamente decide autorizar (Slack, Lineal, Notion, etc.).



Octokit tiene una limitación: sólo podemos enviar hasta seis palabras como parámetros de búsqueda al punto final que recupera PRs. Por lo tanto, usando alguna intuición mecánica, eliminamos las palabras genéricas, duplicadas y detenemos. Ejemplos de palabras de stop-words incluyen *y*, *para*, etc... así como palabras como *desarrollo*, *eliminado* y "*palabras*" usadas en plantillas como *[x]* entre otras.

-

Un área de mejora aquí es seleccionar estas seis palabras de manera más efectiva. Podría un LLM ayudar en esto? Tal vez deberíamos considerar una frecuencia heurística como la palabra. Además, vale la pena debatir si el *lgtn* debe ser clasificado como una llaga. A pesar de todo, este enfoque nos proporciona una serie de relaciones públicas que comparten cierto contexto con la nueva PR. A continuación, tenemos que ordenarlos por relevancia.

-

Hemos considerado varios heurísticos, como el número de cambios de línea o líneas añadieron, pero esto puede ser engañoso por varias razones. Por ejemplo, mover un archivo podría resultar en muchos cambios de línea sin significar una alteración importante. Además, el gran volumen de cambios de línea no siempre refleja la importancia del cambio.

-

En cuanto a la fecha de las relaciones públicas, hemos observado discrepancias significativas entre los equipos. Las opiniones varían: algunos creen que las relaciones públicas más antiguas proporcionan un contexto más relevante, mientras que otros argumentan que las relaciones públicas más nuevas son más pertinentes.

-

En última instancia, nos hemos arreglado sobre el número de comentarios como nuestra métrica primaria. El número de comentarios en una PR se correlaciona no sólo con la cantidad de contexto proporcionado sino también con el alcance del debate en torno a la lógica empresarial, que es crucial para que Watermelon indexe. Las relaciones públicas que implican algo más que respuestas '*lgtn*' son más beneficiosas para nuestros propósitos.

-

Aunque la latencia se cita a menudo como la principal limitación para las aplicaciones impulsadas por RAG, no es nuestra principal preocupación. La revisión de código puede llevar mucho tiempo en términos computacionales, y no necesita ser una operación en tiempo real. Somos capaces de responder en pocos segundos, un proceso que normalmente toma horas o incluso días para recibir una respuesta inicial. Nuestra restricción más apremiante es la limitación de la tasa de GitHub-s API (Octokit), que requiere que respondamos en no más de 3 segundos. Si esto se convierte en un problema, tenemos una estrategia para ejecutar el algoritmo en un hilo separado, pero eso no es una prioridad en este momento.

-

El desafío técnico en esta parte del flujo es la precisión. No necesariamente debido a la alucinación, después de todo [el caso de uso asesino para LLMs es la sumación](#), sino porque construir un algoritmo de búsqueda es realmente difícil.

-

Lo que significa que tenemos que ejecutar un paso adicional al golpear ciertas API. Por ejemplo, al golpear la API de Jira tenemos que ejecutar una consulta de JQL que busca problemas que contienen las palabras aleatorias ya sea en su resumen o descripción, y ordenarlos por fecha descendente para mejorar la precisión

-

[sandía/utils/acciones/getJira.ts](#)

```
const cleanRandomWords = Array.from(
  new Set(randomWords?.map((word) => removeSpecialChars(word)))
);
const summaryQuery = cleanRandomWords
  .map((word) => `summary ~ "${word}"`)
  .join(" OR ");
const descriptionQuery = cleanRandomWords
  .map((word) => `description ~ "${word}"`)
  .join(" OR ");
const jql = `(${summaryQuery}) OR (${descriptionQuery}) ORDER BY created DESC`;
```

```
const cleanRandomWords = Array.from(
  new Set(randomWords?.map((word) => removeSpecialChars(word)))
);
const summaryQuery = cleanRandomWords
  .map((word) => `summary ~ "${word}"`)
  .join(" OR ");
const descriptionQuery = cleanRandomWords
  .map((word) => `description ~ "${word}"`)
  .join(" OR ");
const jql = `(${summaryQuery}) OR (${descriptionQuery}) ORDER BY created DESC`;
```

Análisis de código estético simplificado

La detección de olores de código y la seguridad de las vulnerabilidades también es parte de un [proceso eficaz](#) de [revisión](#) de . Comenzando con la detección de registros de consola (y sus equivalentes en varios lenguajes de programación importantes), Watermelon comenta la línea PR diffs cada vez que se detecta un error. Capacidades ampliadas para identificar una gama más amplia de errores pronto.

-

Para ello corremos 2 subpasos:

-

Primero, ignoramos los comentarios. Son líneas que comienzan con // o " (con comentarios multi-línea siendo nuestro punto de fallo).

-

Luego, analizamos la diferencia de la PR a través de un LLM con el siguiente aviso:

[sandía/utils/actions/detectConsoleLogs.ts](#)

```
const consoleLogDetectionPrompt = `This is a list of code additions. Identify
if there's a console log or its equivalent in another programming language
such as Java, Golang, Python, C, Rust, C++, Ruby, etc.
(console.log(), println(), println!(), System.out.println(), print(), fmt.Println(), puts, and cout
<< "Print a String" << endl; are some examples).
If the console log or its equivalent in another language is in a code comment, don't
count it as a detected console log. For example JavaScript comments start with // or /*,
Python comments start with #.
Other console functions such as console.info() shouldn't be counted as console logs.
Ignore code comments from this analysis.
Something like 'input[type="email"]' is fine and should not be counted as a console log.
If there is a console log, return "true", else return "false".
If you return true, return a string that that has 2 values: result (true) and the line of code.
The line value, is the actual line in the file that contains the console log.
For example: true,console.log("hello world");`;
```

```
const consoleLogDetectionPrompt = `This is a list of code additions. Identify
if there's a console log or its equivalent in another programming language
such as Java, Golang, Python, C, Rust, C++, Ruby, etc.
(console.log(), println(), println!(), System.out.println(), print(),
fmt.Println(), puts, and cout << "Print a String" << endl; are some examples).
If the console log or its equivalent in another language is in a code comment,
don't
count it as a detected console log. For example JavaScript comments start with //
or /*,
Python comments start with #.
Other console functions such as console.info() shouldn't be counted as console
logs.
Ignore code comments from this analysis.
Something like 'input[type="email"]' is fine and should not be counted as a console
log.
If there is a console log, return "true", else return "false".
If you return true, return a string that that has 2 values: result (true) and the
line of code.
The line value, is the actual line in the file that contains the console log.
For example: true,console.log("hello world");`;
```

Para que sea preciso tenemos que construir un RegEx por idioma para detectar registros de consola. Podría funcionar si invertimos las horas en construir muy buenos RegExes. Sin embargo, analizar el código en la línea diffs de una PR con un LLM podría permitirnos hacer

cosas más ambiciosas en el futuro, como permitirnos comparar con más precisión la intención con la implementación (más adelante en esta entrada de blog).

-

[El centiter de árboles](#) también entra en juego en nuestra visión. No sólo nos permitiría filtrar los comentarios multilínea muy fácilmente, sino que nos permitiría entender mejor lo que una línea diff está haciendo desde un punto de vista semántico mediante el análisis de la PR.s AST (Árbol de Sintaxis Revisión). Una vez más, esto también podría permitirnos comparar la intención con la implementación de una mejor manera.

-

También podría ser un complemento de las técnicas. Este flujo sigue siendo muy exploratorio.

-

El paso final: Pre-revisación de relaciones públicas y darles un primer chequeo de salud

El propósito de esta característica es racionalizar el proceso de revisión de código. Su objetivo es ayudar a los desarrolladores a identificar qué relaciones públicas necesitan una revisión más exhaustiva y alentar comentarios más significativos que simplemente lgtn.

-

Actualmente, nos estamos centrando en dos áreas clave: comparar la intención declarada de PR con su implementación real, y detectar registros de consola. Asignamos una calificación de 1 a 10 a cada PR en función de estos criterios. Basándonos en la partición, clasificamos las relaciones públicas como "Don't Merge", "Take a Deeper Dive" o "Safe to Merge".

-

Evaluamos la alineación de la intención con la implementación comparando semánticamente el título de PR, que representa la intención, con el contenido agregado de todos los compromisos de PR, que constituye la implementación.

-

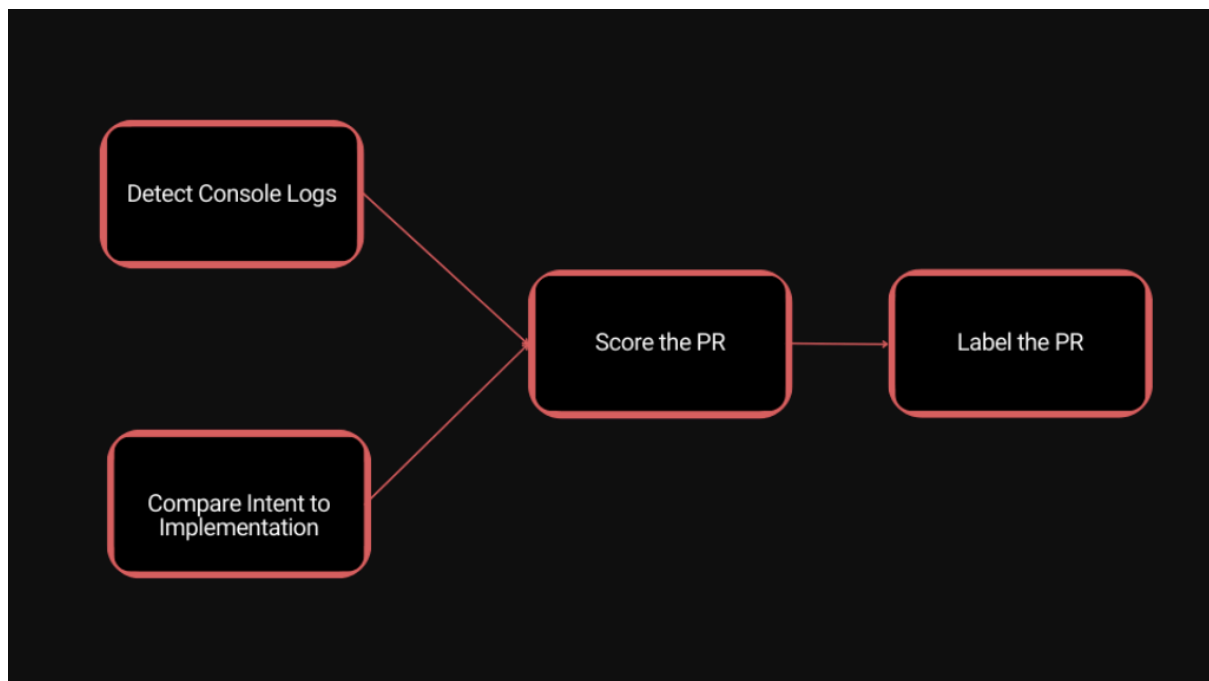
[sandía/utls/acciones/etiquetaPullRequest.ts](#)

```
let prompt = `The goal of this PR is to: ${prTitle}. \n The information
related to this PR is: ${businessLogicSummary}. \n On a scale of 1(very
different)-10(very similar), how similar the PR's goal and the PR's related
information are? Take into account semantics. Don't explain your reasoning,
just print the rating. Don't give a range for the rating, print a single
value.`;
```

```
let prompt = `The goal of this PR is to: ${prTitle}. \n The information related to this PR is: ${businessLogicSummary}. \n On a scale of 1(very different)-10(very similar), how similar the PR's goal and the PR's related information are? Take into account semantics. Don't explain your reasoning, just print the rating. Don't give a range for the rating, print a single value.`;
```

Esta es una zona emergente para nosotros. Como se mencionó anteriormente, al analizar el AST de PR. prevemos avanzar significativamente esta capacidad.

-



Pronto venga.

Como grandes creadores de [código abierto](#) queremos apoyar la capacidad de manejar Watermelon con un LLM de código abierto. Llama y Mistral son candidatos que echamos un vistazo, mientras que a más largo plazo también queremos apoyar una versión auto-anfitriona de Watermelon que implicaría que también incluiríamos apoyo para Ollama.

-

También queremos ayudar a las empresas a hacer algo más que detectar registros de consolas para mantener su cumplimiento SOC 2. Por ejemplo, ayudar a las empresas a no empujar los datos de PII a la producción. Hay mucho más para comentar que un lgtm en una PR después de todo ;)

-

Para mejorar la forma en que apuntamos y etiquetamos las relaciones públicas también queremos traer el comportamiento del autor. Queremos medir la relación de comentario / aprobación de relaciones RR. Una autora de relaciones públicas que obtiene altas

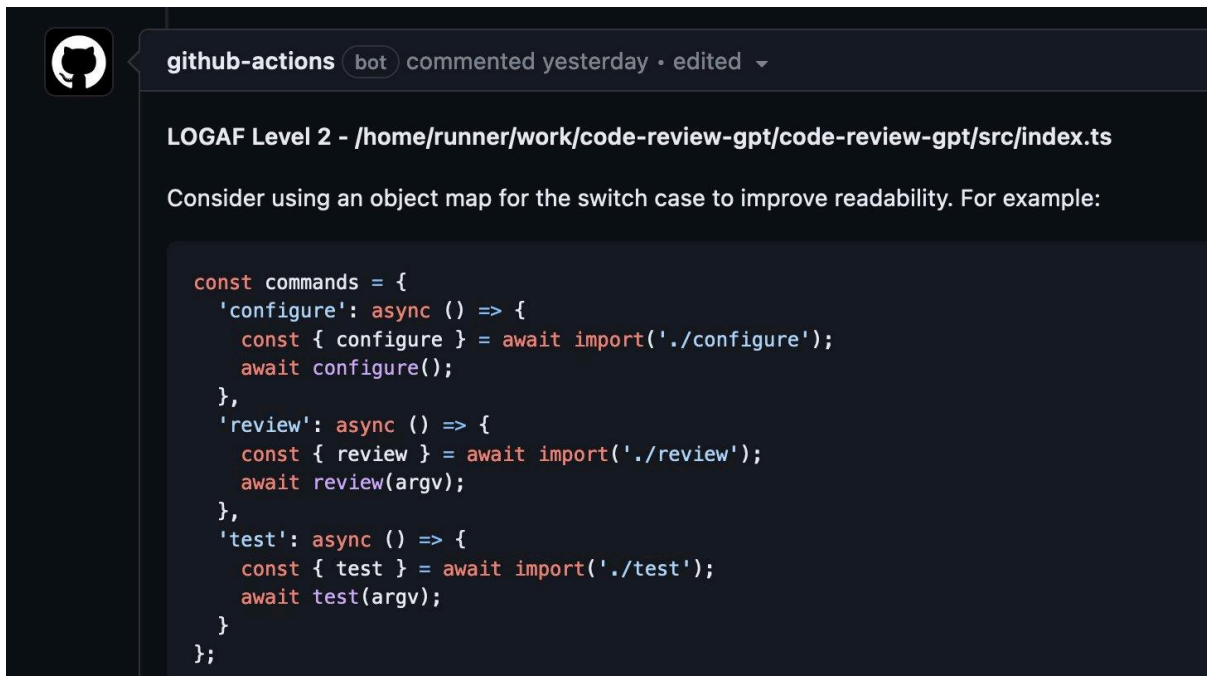
puntuaciones en esta proporción es de algunos que está revisando las relaciones públicas, y compartiendo contexto a través de sus comentarios también (un lgtn no cuenta), genial. Lo que esta persona envía tiene una mayor posibilidad de cumplir con los requisitos de negocio. ¿La persona no está aprobando relaciones públicas? Esa persona no está haciendo revisión de código, por lo tanto no es necesario aprender acerca de la base de código y la puntuación para una sesión de relaciones públicas por parte de esa persona debe ser castigado. Alto número de aprobaciones, pero sin comentarios? La persona es sólo estampación de goma (quizás con un lgtn), y tal puntuación también debería ocurrir.

-

Agente revisor de código personal impulsado por LLM.

Se ejecuta en un pipeline CI/CD.

Proporciona sugerencias súper detalladas y útiles sobre el código, en base al entrenamiento de estándares de desarrollo pero con la capacidad de hacer ingesta de los definidos internamente por la organización los cuales pueden llegar a ser bastante espaciales y son los más relevantes a tomar en cuenta.



```
github-actions bot commented yesterday · edited ▾  
  
LOGAF Level 2 - /home/runner/work/code-review-gpt/code-review-gpt/src/index.ts  
  
Consider using an object map for the switch case to improve readability. For example:  
  
const commands = {  
  'configure': async () => {  
    const { configure } = await import('./configure');  
    await configure();  
  },  
  'review': async () => {  
    const { review } = await import('./review');  
    await review(argv);  
  },  
  'test': async () => {  
    const { test } = await import('./test');  
    await test(argv);  
  }  
};
```

Se obtienen todos los archivos modificados usando el comando git:

```
`git diff --solo nombre --diff-filter=AMT ${baseSha} ${githubSha}`
```

Y crea un mensaje personalizado para decirle al LLM exactamente qué hacer

Esta prueba de concepto se usa un *Agente revisor de código personal impulsado por LLM* que se basa en **Lang ChainAI** concretamente para realizar solicitudes al LLM.
<https://github.com/langchain-ai/langchain>

En este caso, OpenAI [GPT-4](#) funciona bastante bien y ahora está disponible ya que los modelos de la primera camada son legacy y están depreciados:

InstructGPT models			
SHUTDOWN DATE	LEGACY MODEL	LEGACY MODEL PRICE	RECOMMENDED REPLACEMENT
2024-01-04	text-ada-001	\$0.0004 / 1K tokens	gpt-3.5-turbo-instruct
2024-01-04	text-babbage-001	\$0.0005 / 1K tokens	gpt-3.5-turbo-instruct
2024-01-04	text-curie-001	\$0.0020 / 1K tokens	gpt-3.5-turbo-instruct
2024-01-04	text-davinci-001	\$0.0200 / 1K tokens	gpt-3.5-turbo-instruct
2024-01-04	text-davinci-002	\$0.0200 / 1K tokens	gpt-3.5-turbo-instruct
2024-01-04	text-davinci-003	\$0.0200 / 1K tokens	gpt-3.5-turbo-instruct
Pricing for the replacement <code>gpt-3.5-turbo-instruct</code> model can be found on the pricing page .			
Base GPT models			
SHUTDOWN DATE	LEGACY MODEL	LEGACY MODEL PRICE	RECOMMENDED REPLACEMENT
2024-01-04	ada	\$0.0004 / 1K tokens	babbage-002
2024-01-04	babbage	\$0.0005 / 1K tokens	babbage-002
2024-01-04	curie	\$0.0020 / 1K tokens	davinci-002
2024-01-04	davinci	\$0.0200 / 1K tokens	davinci-002
2024-01-04	code-davinci-002	---	gpt-3.5-turbo-instruct
Pricing for the replacement <code>babbage-002</code> and <code>davinci-002</code> models can be found on the pricing page .			

LangChain es un marco de trabajo de inteligencia artificial que facilita el desarrollo de aplicaciones basadas en lenguaje. Proporciona abstracciones flexibles y un conjunto de herramientas orientado a la inteligencia artificial para construir aplicaciones que utilizan Modelos de Lenguaje de Gran Tamaño (**LLM**, por sus siglas en inglés) de manera consciente del contexto y con capacidad de razonamiento.

Un **LLM** es un modelo de lenguaje que consta de una red neuronal con muchos parámetros (normalmente miles de millones o más), entrenados en grandes cantidades de texto sin etiquetar mediante aprendizaje autosupervisado o aprendizaje semi supervisado. Los **LLM** son modelos de propósito general que se destacan en una amplia gama de tareas, en lugar de estar capacitados para una tarea específica.

Por lo tanto, nos permite como desarrolladores construir aplicaciones que utilizan **LLM** para diversas tareas como chatbots, generación de preguntas y respuestas, resúmenes, análisis de documentos, búsqueda personalizada y más.

Con la nueva herramienta [LangSmith](#) Lang Chain AI, obtienes este increíble panel.

Realice un seguimiento de sus costos y respuestas de manera muy sencilla.

default

P50 Latency: 8.54sP99 Latency: 71.18sTotal Tokens: 143,251

TracesAll RunsSetup

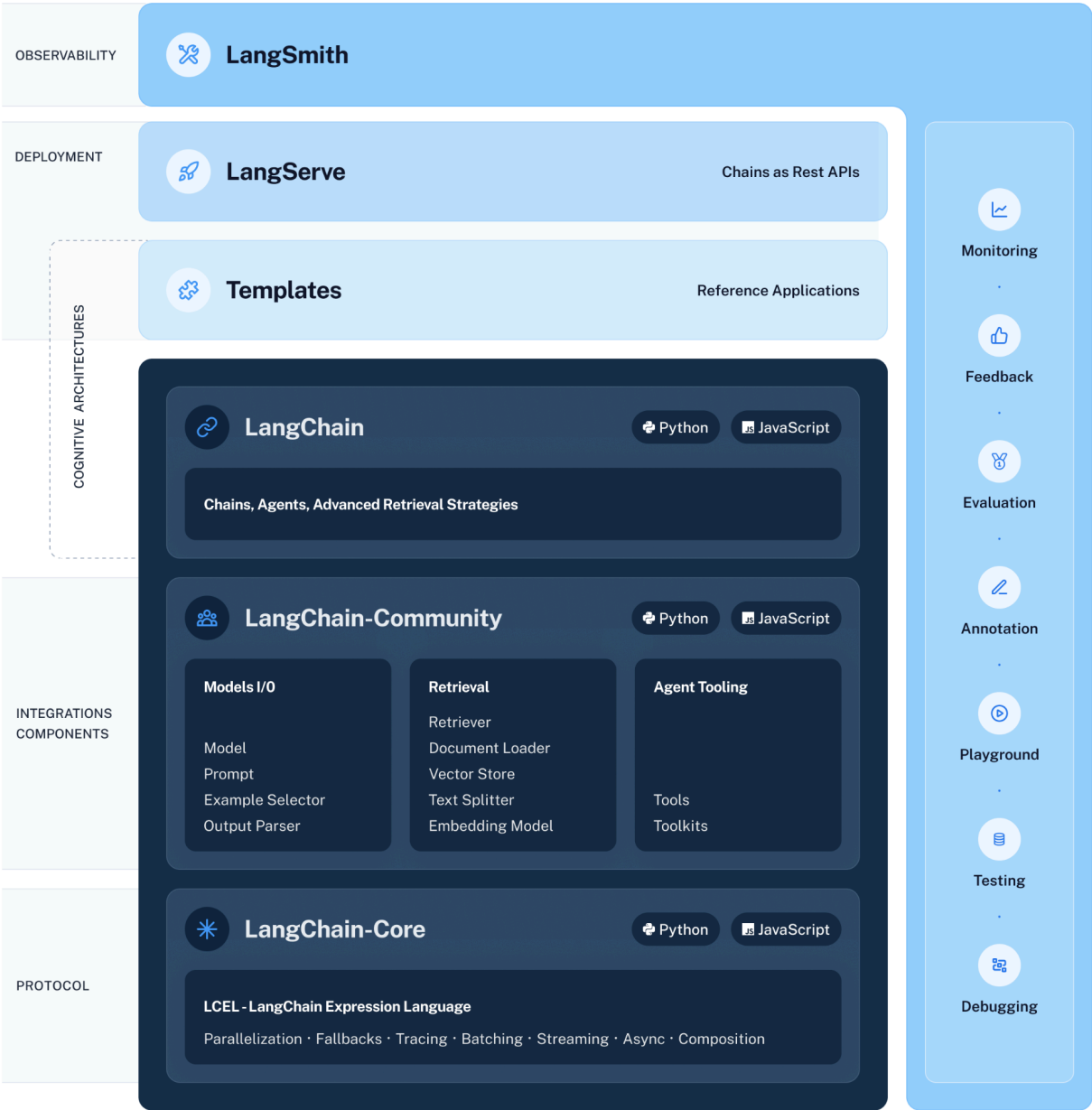
eg. eq(run_type, "chain")

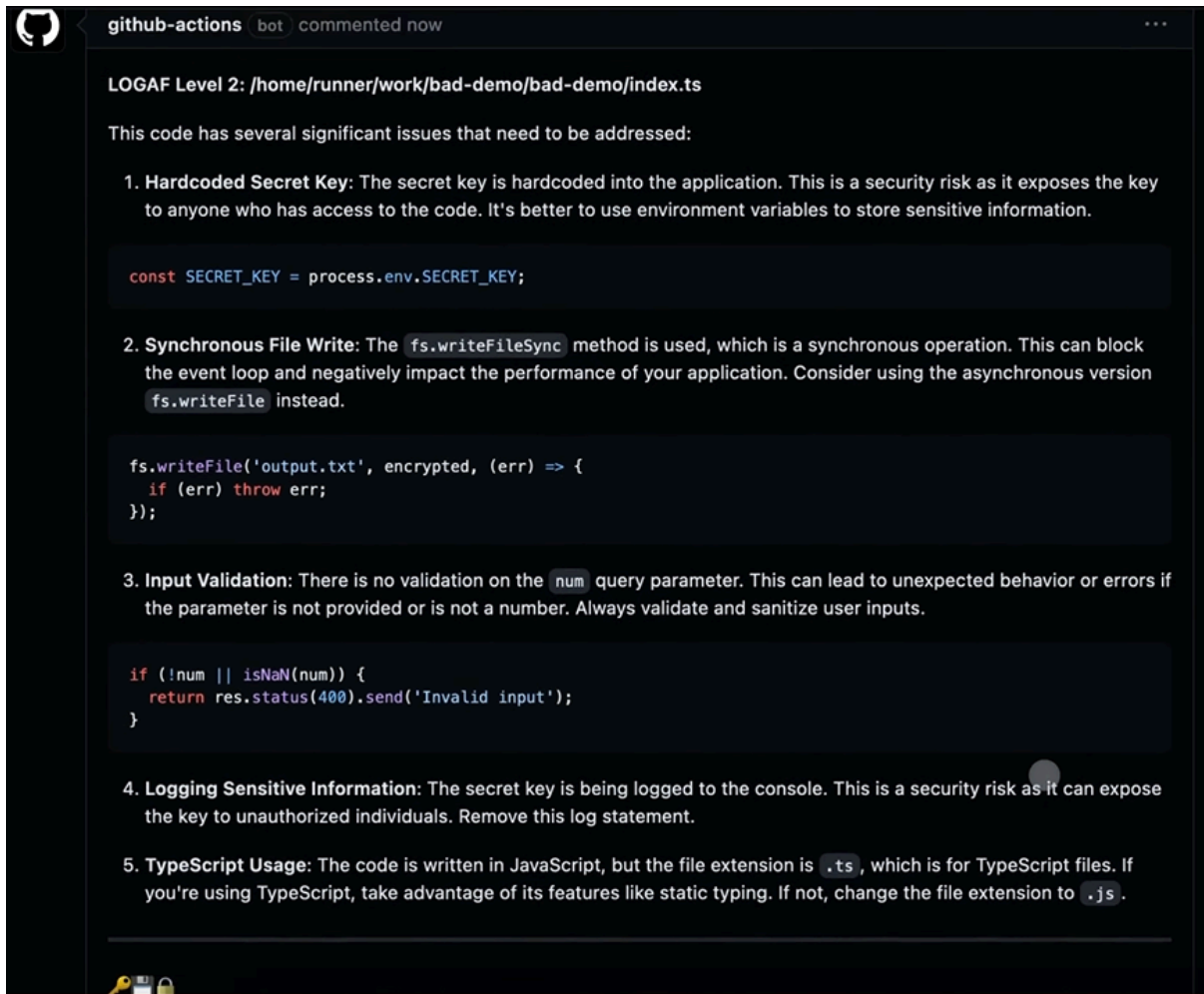
Columns

<input type="checkbox"/>	Run Type	Name	Input	Start Time	Latency	Tokens	Tags
<input type="checkbox"/>	LLM	OpenAIChat	You are a senior de...	20/07/2023, 11:56:00	1.27s	246	
<input type="checkbox"/>	LLM	OpenAIChat	You are a senior de...	20/07/2023, 11:55:42	17.81s	612	
<input type="checkbox"/>	LLM	OpenAIChat	As a senior develop...	20/07/2023, 11:55:28	13.88s	742	
<input type="checkbox"/>	LLM	OpenAIChat	You are a senior de...	20/07/2023, 11:52:07	1.18s	255	
<input type="checkbox"/>	LLM	OpenAIChat	You are a senior de...	20/07/2023, 11:51:48	18.98s	630	
<input type="checkbox"/>	LLM	OpenAIChat	As a senior develop...	20/07/2023, 11:51:34	13.88s	751	

Agente revisor de código personal impulsado por LLM luego califica las respuestas y publica los problemas más graves en su PR

<https://smith.langchain.com>





LOGAF: /home/runner/work/bad-demo/bad-demo/index.ts

Este código tiene varios problemas importantes que deben abordarse:

1. Clave secreta codificada: la clave secreta está codificada en la aplicación. Este es un riesgo de seguridad ya que expone la clave.

a cualquiera que tenga acceso al código. Es mejor utilizar variables de entorno para almacenar información confidencial.

```
const SECRET_KEY = proceso.env.SECRET_KEY;
```

2. Escritura de archivos sincrónica: se utiliza el método `fs.writeFileSync`, que es una operación sincrónica. Esto puede bloquear el bucle de eventos e impactar negativamente el rendimiento de su aplicación. Considere usar la versión asincrónica `fs.writeFile` en su lugar.

```
fs.writeFile('salida.txt', cifrado, (err) {
```

```
  si (error)
```

```
    tirar error;
```


3. Validación de entrada: no hay validación en el parámetro de consulta numérico. Esto puede provocar comportamientos inesperados o errores si

el parámetro no se proporciona o no es un número. Siempre valide y desinfecte las entradas de los usuarios.


```
if (!num || esNaN(num)) {
```

```
  devolver entrada};
```

4. Registro de información confidencial: la clave secreta se registra en la consola. Este es un riesgo de seguridad que puede exponer la clave para personas no autorizadas. Elimine esta declaración de registro.
5. Uso de TypeScript: el código está escrito en JavaScript, pero la extensión del archivo es .ts, que es para archivos TypeScript. Si estás utilizando TypeScript, aprovecha sus funciones como la escritura estática. Si no, cambie la extensión del archivo a .js

 Merged

Close new brain modal after creation #720
gozineb merged 1 commit into `main` from `ux-improvement-2` 6 minutes ago

 **github-actions** bot commented 29 minutes ago · edited

LOGAF Level 3: `/home/runner/work/quivr/quivr/frontend/lib/components/NavBar/components/NavItems/components/BrainsDropDown/components/AddBrainModal.tsx`




The code is generally good, but there are a few areas that could be improved:

- Error handling:** The error handling could be improved by creating a custom error handler function. This would make the code more readable and maintainable. For example:

```
const handleError = (err: any) => {  
  let message = JSON.stringify(err);  
  if (axios.isAxiosError(err) && err.response?.status === 429) {  
    message = JSON.stringify((err.response as { data: { detail: string } }).data.detail);  
  }  
  publish({ variant: "danger", text: message });  
};
```

Then, in your `catch` block, you can simply call `handleError(err)`.

- Form validation:** Consider using a form validation library like Formik or react-hook-form. This would make your form validation more robust and easier to manage.
- State management:** Consider using a reducer for managing the state. This would make your state management more predictable and easier to debug.
- Code comments:** Consider adding comments to your code to explain what each part of the code does. This would make your code easier to understand for other developers.

Powered by [Code Review GPT](#)