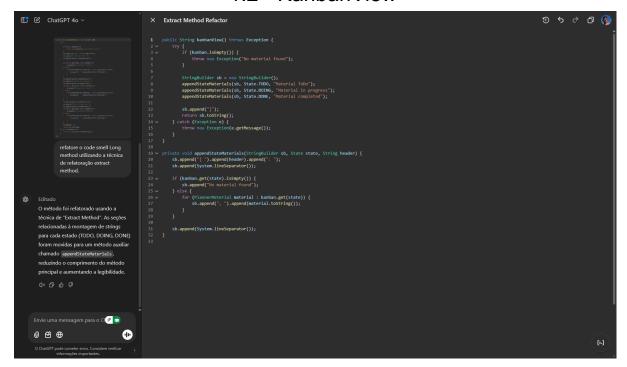
Qualidade de Software - Refatoração de Code Smells

Lorena Emanuele Pinheiro Ferreira - 516293 Robson José Dias Venancio - 508487

1- Long Method

1.1 - LeitnerSystem

1.2 - KanbanView



1.3 - TodoTracker

1.4 - StudyGoal

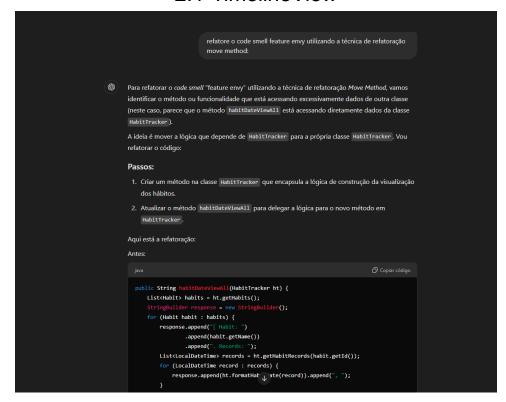
```
× Extract Method Refactor
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      ୭ ଚ ∂ ⊡ 🚱
                                                                                                                                                                                                        1 > public String setGoalSummary() {
2    StringBuilder summary = new StringBuilder();
3    appendHeader(summary, "Goal Summary:");
                                                                                                                                                                                                                     this.summary = summary.toString();
return summary.toString();
}
A refatoração reduziu o tamanho do
                         linhas ainda pode ser otimizado para
                          atender à regra NCSS. Vamos ajustar
ainda mais o código usando métodos
                         auxiliares para agrupar lógica repetida e reduzir a contagem de linhas no método
                          Vou refatorar o código.
                                                                                                                                                                                                                  v private void appendCollection!fNotNull(stringBullder summary, String title, CollectionCstring> collection) {
    if (collection != null) {
        summary.append(title).append("\n");

        for (String item : collection) {
            summary.append(title).append("\n");
        }
            remains.append(title).append("\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).append(\n");
            remains.append(title).ap
                          contagem de linhas no método principal
                            setGoalSummary . As verificações
condicionais foram simplificadas usando
                              métodos auxiliares genéricos como appendConditionalSection ,
                                                                                                                                                                                                                v private void append0bjectfNotNull(StringBullder summary, String title, Object obj) {
v          if (obj != null) {
               appendSection(summary, title, obj.toString());
}
                              appendCollectionIfNotNull, e
                            ainda mais a legibilidade e
                  Envie uma mensagem para o Cl 🕖 👛
                 0 ≝ ⊕
```

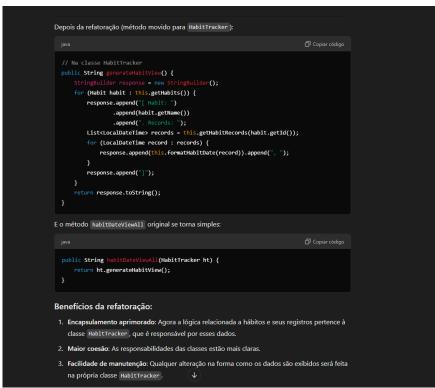
1.5 - StudyMaterial

2 - Feature Envy

2.1-TimelineView



```
Antes:
                                                                                  Copiar código
  public String habitDateViewAll(HabitTracker ht) {
     List<Habit> habits = ht.getHabits();
      StringBuilder response = new StringBuilder();
      for (Habit habit : habits) {
          response.append("[ Habit: ")
                 .append(habit.getName())
                  .append(". Records: ");
          List<LocalDateTime> records = ht.getHabitRecords(habit.getId());
          for (LocalDateTime record : records) {
              response.append(ht.formatHabitDate(record)).append(", ");
          response.append("]");
      return response.toString();
Depois da refatoração (método movido para HabitTracker ):
                                                                                  Copiar código
 public String generateHabitView() {
   StringBuilder response = new StringBuilder();
      for (Habit habit : this.getHabits()) {
         response.append("[ Habit: ")
                 .append(habit.getName())
         .append(". Records: ");
List<LocalDateTime> records = this.getHabitRecords(habit.getId());
          for (LocalDateTime record : records) {
              response.append(this.formatHabitDate(record)).append(", ");
          response.append("]");
      return response.toString();
```



2.2 - GeneralSearch

Soluções Aplicadas:

- 1. Delegação da Lógica de Logging para SearchLog:
 - Foi identificado que a lógica relacionada ao registro do histórico (addSearchHistory),
 contagem de usos (setNumUsages), e criação da mensagem de log (getLogName) estava
 melhor localizada na classe SearchLog .
 - Criado o método logSearchAndReturnMessage em SearchLog para encapsular essa lógica.

```
public class SearchLog {
   public String logSearchAndReturnMessage(String text) {
      this.addSearchHistory(text);
      this.setNumUsages(this.getNumUsages() + 1);
      return "Logged in: " + this.getLogName();
   }
}
```

2. Simplificação do Método handleSearch:

- O método handleSearch foi modificado para delegar as responsabilidades relacionadas ao
 SearchLog .
- A lógica de busca permaneceu focada nas chamadas aos gerenciadores, mas a responsabilidade de log foi transferida.

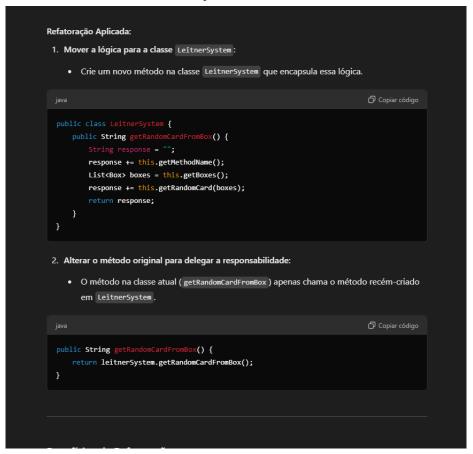
Refatorado:

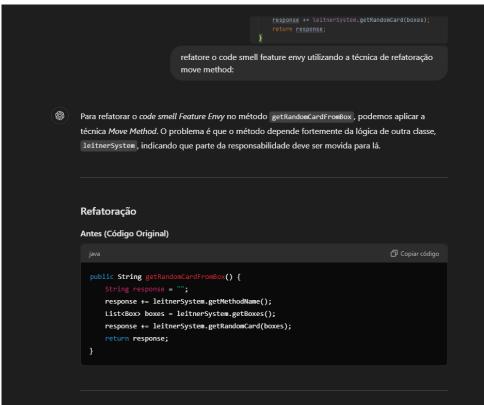
```
private List<String> handleSearch(String text) {
   List<String> results = new ArrayList<>();
   results.addAll(CardManager.getCardManager().searchInCards(text));
   results.addAll(HabitTracker.getHabitTracker().searchInHabits(text));
   results.addAll(TodoTracker.getInstance().searchInTodos(text));
   results.addAll(StudyMaterial.getStudyMaterial().searchInMaterials(text));
   results.addAll(StudyTaskManager.getStudyTaskManager().searchInRegistries(text));

// Delegando o logging para SearchLog
   results.add(this.searchLog.logSearchAndReturnMessage(text));

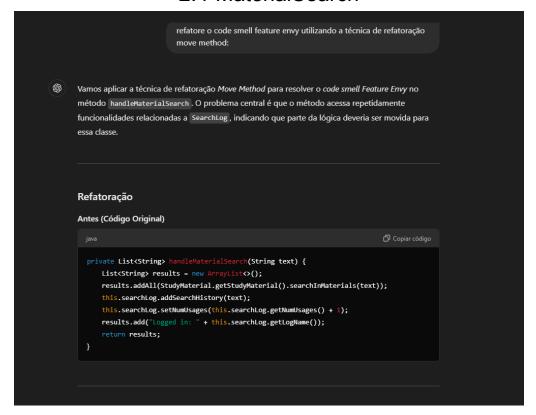
return results;
}
```

2.3 - StudyCardsController



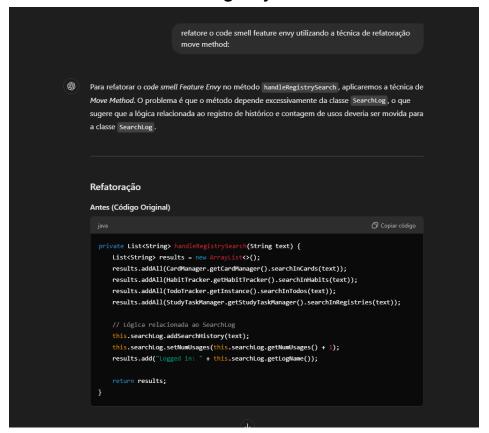


2.4-MaterialSearch



Refatoração Aplicada: 1. Mover a lógica relacionada ao SearchLog para a própria classe SearchLog: • Adicione um novo método na classe SearchLog que encapsula as operações relacionadas ao registro de busca. Copiar código public String logSearch(String text) { this.addSearchHistory(text); this.setNumUsages(this.getNumUsages() + 1); return "Logged in: " + this.getLogName(); 2. Modificar o método handleMaterialSearch para delegar a responsabilidade de logging: • Agora o método handleMaterialSearch se torna mais simples, focando apenas na lógica Copiar código private List<String> handleMaterialSearch(String text) { List<String> results = new ArrayList<>(); results. add All (Study Material.get Study Material (). search In Materials (text));results.add(this.searchLog.logSearch(text)); // Delegação para SearchLog return results;

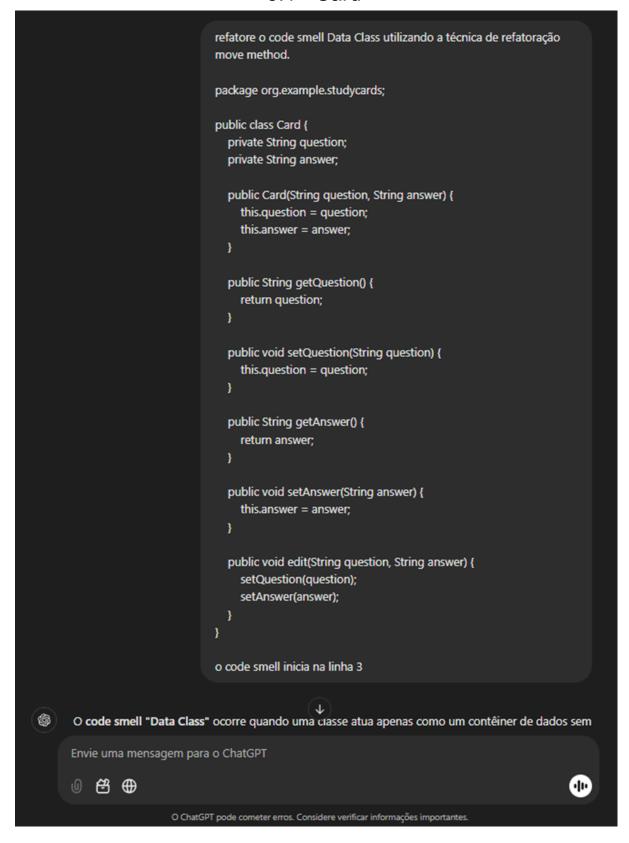
2.5 - RegistrySearch



Refatoração Aplicada: 1. Mover a lógica de registro de busca para SearchLog: • Adicione um novo método na classe SearchLog para encapsular a lógica relacionada ao histórico e contagem de usos. Copiar código public class SearchLog { public String logSearch(String text) { this.addSearchHistory(text); this.setNumUsages(this.getNumUsages() + 1); return "Logged in: " + this.getLogName(); 2. Alterar o método handleRegistrySearch para delegar a responsabilidade ao SearchLog; private List<String> handleRegistrySearch(String text) { List<String> results = new ArrayList<>(); results. add All (Card Manager.get Card Manager().search In Cards (text));results.addAll(HabitTracker.getHabitTracker().searchInHabits(text)); results.addAll(TodoTracker.getInstance().searchInTodos(text)); results.addAll(StudyTaskManager.getStudyTaskManager().searchInRegistries(text)); results.add(this.searchLog.logSearch(text)); return results;

3 - Data Class

3.1 - Card



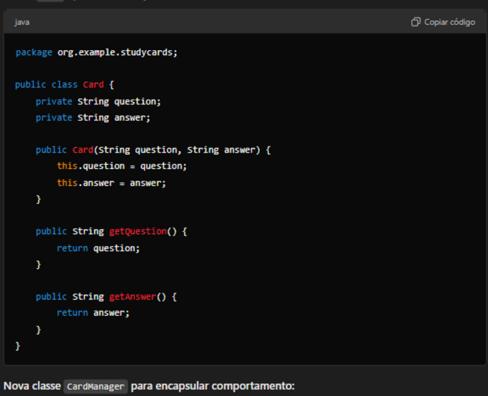
\$

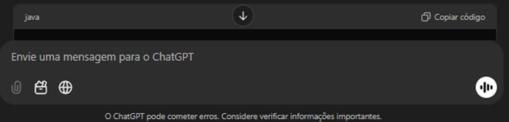
O code smell "Data Class" ocorre quando uma classe atua apenas como um contêiner de dados sem comportamento relevante. Para resolver isso, podemos usar a técnica de Move Method, onde os métodos que fazem sentido serem implementados em outra classe são movidos para lá, e a classe original deixa de ser apenas um repositório de dados.

A seguir, vamos identificar o comportamento da classe card e sugerir uma refatoração. Neste caso, a responsabilidade de editar os dados pode ser movida para uma nova classe que manipula os cartões (cardManager , por exemplo). Vamos refatorar:

Código Refatorado:

Classe card após a refatoração:





32 - Reference

```
public void setViewCount(int viewCount) {
    this.viewCount = viewCount;
  public int getDownloadCount() {
    return downloadCount;
  public void setDownloadCount(int downloadCount) {
    this.downloadCount = downloadCount;
  public int getShareCount() {
    return shareCount;
  public void setShareCount(int shareCount) {
    this.shareCount = shareCount;
refatore o code smell Data Class method utilizando a técnica de
refatoração Move Method
```

Para refatorar o código utilizando a técnica de refatoração Move Method, o objetivo é mover métodos que manipulam dados de uma classe para uma classe que seja mais apropriada para esse tipo de operação. Como a classe Reference está sendo usada para representar dados, podemos considerar que os métodos de manipulação de dados podem ser movidos para uma classe que trate desses dados, talvez uma classe especializada, para melhorar a coesão da classe Reference e diminuir sua responsabilidade.

Aqui está um exemplo de como mover os métodos de acesso (getters e setters) para uma nova classe ReferenceData:

Refatoração

\$

- 1. Criação da Classe ReferenceData: A nova classe ReferenceData será responsável por armazenar e manipular os dados relacionados à Reference .
- 2. Movendo os métodos: Os métodos set e get podem ser movidos para a nova classe ReferenceData enquanto a classe Reference continua com as propriedades.

Código Refatorado

Envie uma mensagem para o ChatGPT









Código Refatorado Classe ReferenceData Copiar código package org.example.studymaterial; public class ReferenceData { private String title; private String description; private String link; private String accessRights; private String license; private boolean isDownloadable; private int rating; private String language; private int viewCount; private int downloadCount; private int shareCount; // Getters and Setters public void setTitle(String title) { this.title = title; } public String getTitle() { return title; } public void setDescription(String description) { this.description = description; } public String getDescription() { return description; } public void setLink(String link) { this.link = link; public String getLink() { return link; \downarrow } Envie uma mensagem para o ChatGPT 0 ≝ ⊕

3.3 - Todo

```
refatore o code smell data class utilizando a técnica de refatoração
                                 move method.
6
      Para refatorar o code smell "Data Class" utilizando a técnica de Move Method, você pode transferir
      responsabilidades para métodos que utilizem os atributos diretamente na própria classe Todo . Isso
      faz com que ela seja mais coesa e encapsule comportamentos relacionados aos dados que ela
      armazena.
      No código apresentado, a classe ToDo parece estar funcionando apenas como uma estrutura de
      dados com getters e setters. Vamos transformá-la em uma classe mais orientada a comportamentos
      ao adicionar métodos que operem diretamente nos atributos dela.
      Aqui está a refatoração:
      Refatoração do Código:
                                                                                        (T) Copiar código
         package org.example.studyPlanner;
         import java.text.MessageFormat;
         public class ToDo implements PlannerMaterial {
            private Integer id;
            private String title;
            private String description;
            private int priority;
             public ToDo(Integer id, String title, String description, int priority) {
                 this.id = id;
                                                   \downarrow
                 this.title = title;
                this.description = description:
      Envie uma mensagem para o ChatGPT
      0 ≅ ⊕
                                                                                                   1
                         O ChatGPT pode cometer erros. Considere verificar informações important
```

```
this.description = description;
                                                                                 (Copiar código
          this.priority = priority;
      }
      @Override
      public String toString() {
          return MessageFormat.format(
              "[Priority:{3}] ToDo {0}: {1}, {2}", id, title, description, priority);
      }
      // Behavior added for prioritization
      public boolean isHighPriority() {
          return this.priority > 5;
      }
      public void promotePriority() {
          this.priority += 1; // Simple logic to promote the priority
      public void demotePriority() {
          if (this.priority > 0) {
               this.priority -= 1; // Decrease priority, but ensure it doesn't go below 0
          }
      }
      public void updateDescription(String newDescription) {
          this.description = newDescription.trim(); // Update description with sanitization
      }
      // Getters and Setters
      public Integer getId() { return id; }
      public void setId(Integer id) { this.id = id; }
      public String getTitle() { return title; }
      public void setTitle(String title) { this.title = title; }
      public String getDescription() { return description; }
public void setDescription(String description) { this.description = description; }
      public int getPriority() { return priority; }
      public void setPriority(int priority) { this.priority = priority; }
  }
                                             \downarrow
Altoraçãos Enitas:
Envie uma mensagem para o ChatGPT
  23 ⊕
```

3.4 - Task

```
thic String gatHitle() ( return title() {
this voic scripture tring ( title ) ( title - title ) }
the String gatHammighten() ( "volum description")
this voic settlescription() ( title ) ( title ) this description of searchtion() }
                                     refatore o code smell Data Class utilizando a técnica de refatoração
                                     move method.
6
       A classe Task apresentada contém o code smell conhecido como "Data Class", onde a classe é
       apenas um recipiente para dados e não possui comportamento (métodos relevantes). Para eliminar
       esse code smell, podemos mover métodos que realizam comportamentos significativos diretamente
       relacionados à Task para dentro da classe. Por exemplo:
       Etapas para aplicar a refatoração:
         1. Identificar métodos que interagem com Task em outras partes do código.
         2. Mover esses métodos para a classe Task para encapsular melhor o comportamento e respeitar
            o princípio de responsabilidade única.
       Segue a refatoração inicial:
                                                                                                    Copiar código
          java
          package org.example.studyregistry;
          import java.time.LocalDateTime;
          public class Task extends Registry {
              private String title;
              private String description;
              private String author;
              private LocalDateTime date;
               public Task(String title, String description, String author, LocalDateTime date) {
                    this.title = title;
                   this.name = title; // Verificar se 'name' é necessário
       Envie uma mensagem para o ChatGPT
       O 23 ⊕
                                                                                                                 (40)
                            O ChatGPT pode cometer erros. Considere verificar informações importantes.
```

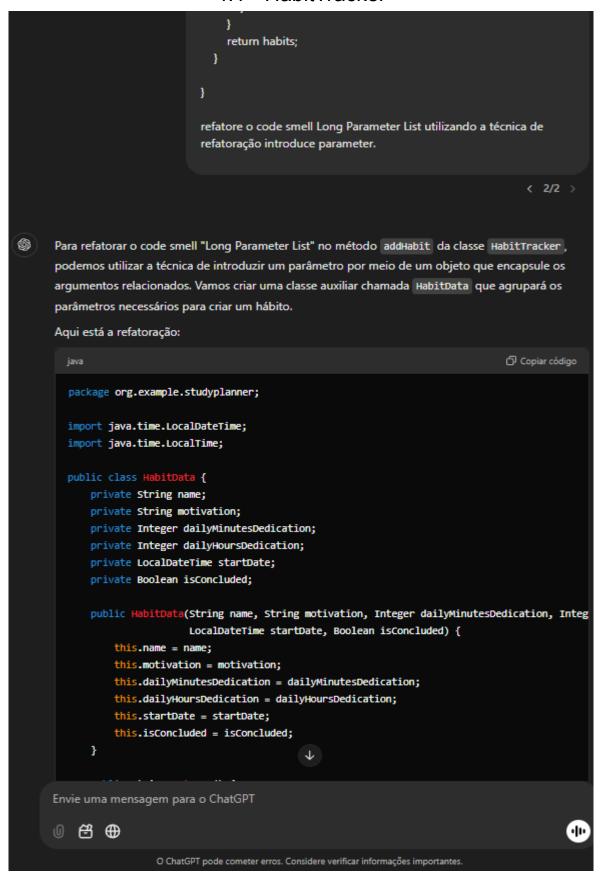
```
(7) Copiar código
// Getters e Setters permanecem apenas se realmente necessários
public String getTitle() {
   return title;
}
public void setTitle(String title) {
   this.title = title;
}
public String getDescription() {
   return description;
public void setDescription(String description) {
    this.description = description;
public String getAuthor() {
   return author;
}
public void setAuthor(String author) {
   this.author = author;
public LocalDateTime getDate() {
   return date;
}
public void setDate(LocalDateTime date) {
    this.date = date;
// Comportamentos adicionados para encapsular lógica dentro da classe
public boolean isOverdue() {
   return LocalDateTime.now().isAfter(this.date);
public String summarizeTask() {
   return String.format("Task: %s | Author: %s | Due: %s", title, author, date);
public void postponeTask(int days) {
    this.date = this.date.plusDays(d(\psi));
```

Envie uma mensagem para o ChatGPT



4 - Long Parameter List

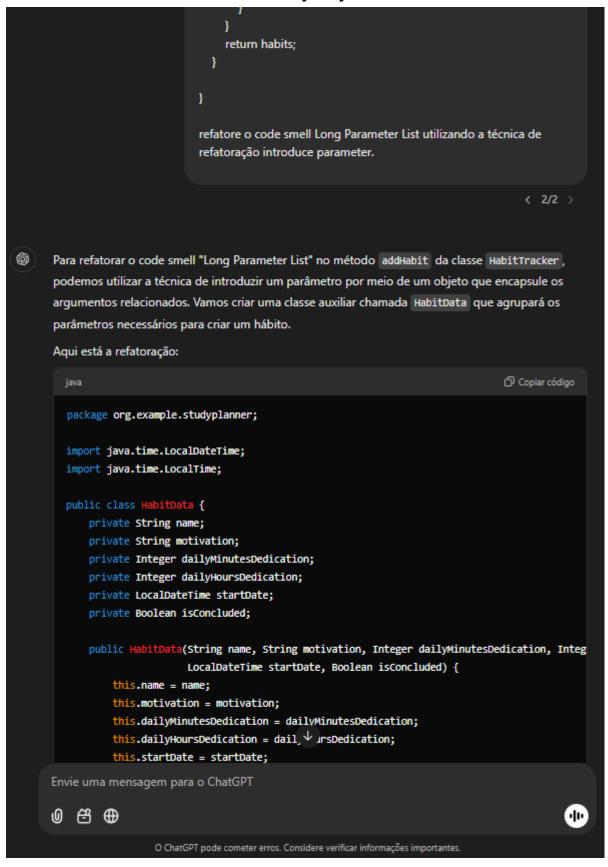
4.1 - HabitTracker



```
public String getName() {
          return name;
      }
      public String getMotivation() {
          return motivation;
      public Integer getDailyMinutesDedication() {
          return dailyMinutesDedication;
      }
      public Integer getDailyHoursDedication() {
         return dailyHoursDedication;
      }
      public LocalDateTime getStartDate() {
         return startDate;
      }
      public Boolean getIsConcluded() {
         return isConcluded;
      }
  }
Refatorando o método addHabit
Agora, alteramos o método addHabit para aceitar uma instância de HabitData em vez de uma
longa lista de parâmetros.
                                                                            Copiar código
  public int addHabit(HabitData habitData) {
      LocalTime dailyDedication = LocalTime.of(habitData.getDailyHoursDedication(), habitDat
      Habit habit = new Habit(habitData.getName(), habitData.getMotivation(), dailyDedicatio
                             habitData.get_artDate(), habitData.getIsConcluded());
      this.habits.add(habit);
      thic tracker out/newt7d
Envie uma mensagem para o ChatGPT
0 23 ⊕
                                                                                       (回)
```

O ChatGPT pode cometer erros. Considere verificar informações importantes.

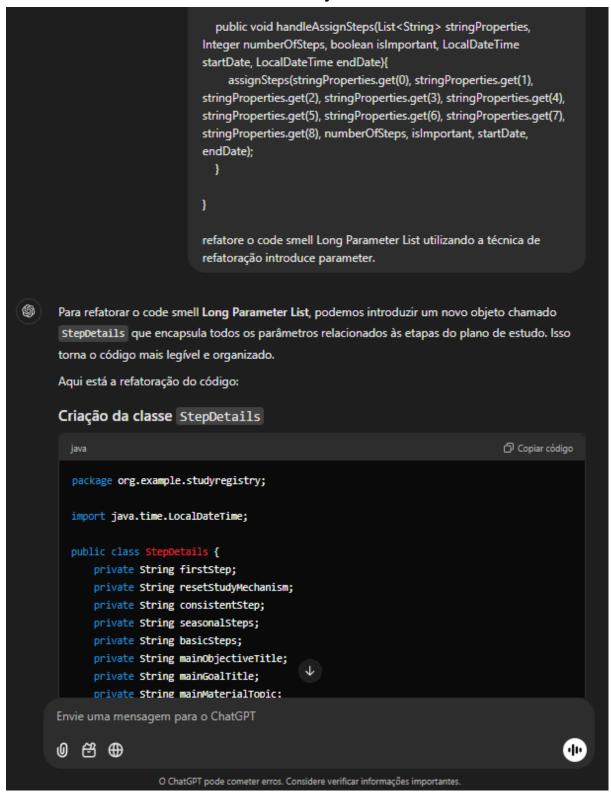
4.2 - StudyObjective



```
this.startDate = startDate;
                                                                             (a) Copiar código
          this.isConcluded = isConcluded;
      }
      public String getName() {
          return name;
      }
      public String getMotivation() {
          return motivation;
      }
      public Integer getDailyMinutesDedication() {
         return dailyMinutesDedication;
      }
      public Integer getDailyHoursDedication() {
          return dailyHoursDedication;
      public LocalDateTime getStartDate() {
          return startDate;
      }
      public Boolean getIsConcluded() {
          return isConcluded;
      }
  }
Refatorando o método addHabit
Agora, alteramos o método addHabit para aceitar uma instância de HabitData em vez de uma
longa lista de parâmetros.
                                                                              Copiar código
  iava
  public int addHabit(HabitData habitData) {
      LocalTime dailyDedication = LocalTime.of(habitData.getDailyHoursDedication(), habitDat
      Habit habit = new Habit(habitData.getName(), habitData.getMotivation(), dailyDedicatio
                              habitData.getStartDate(), habitData.getIsConcluded());
      this.habits.add(habit);
      this.tracker.put(nextId, new ArrayList<>());
      return nextId++;
  }
Envie uma mensagem para o ChatGPT
0 23 ⊕
                                                                                        (III)
```

O ChatGPT pode cometer erros. Considere verificar informações importantes.

4.3 - StudyTask



```
public StepDetails(String firstStep, String resetStudyMechanism, String consistence
                         String mainObjectiveTitle, String mainGoalTitle, String mainMateria
                         Integer numberOfSteps, boolean isImportant, LocalDateTime startDate
          this.firstStep = firstStep;
          this.resetStudyMechanism = resetStudyMechanism;
          this.consistentStep = consistentStep;
          this.seasonalSteps = seasonalSteps;
          this.basicSteps = basicSteps;
          this.mainObjectiveTitle = mainObjectiveTitle;
          this.mainGoalTitle = mainGoalTitle;
          this.mainMaterialTopic = mainMaterialTopic;
          this.mainTask = mainTask;
          this.numberOfSteps = numberOfSteps;
          this.isImportant = isImportant;
          this.startDate = startDate;
          this.endDate = endDate;
      }
      // Getters
      public String getFirstStep() {
          return firstStep;
      }
      public String getResetStudyMechanism() {
         return resetStudyMechanism;
      public String getConsistentStep() {
         return consistentStep;
      }
      public String getSeasonalSteps() {
         return seasonalSteps;
      }
      public String getBasicSteps() {
         return basicSteps;
      public String getMainObjectiveTitle() {
          return mainObjectiveTitle;
      public String getMainGoalTitle() {
Envie uma mensagem para o ChatGPT
0 2 0
                                                                                        (40)
```

```
public String getResetStudyMechanism() {
  return resetStudyMechanism;
}
public String getConsistentStep() {
  return consistentStep;
public String getSeasonalSteps() {
  return seasonalSteps;
public String getBasicSteps() {
   return basicSteps;
}
public String getMainObjectiveTitle() {
  return mainObjectiveTitle;
public String getMainGoalTitle() {
  return mainGoalTitle;
public String getMainMaterialTopic() {
  return mainMaterialTopic;
public String getMainTask() {
  return mainTask;
}
public Integer getNumberOfSteps() {
   return numberOfSteps;
}
public boolean isImportant() {
  return isImportant;
public LocalDateTime getStartDate() {
  return startDate;
}
```

Envie uma mensagem para o ChatGPT



4.4 - Study Plan

