Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

TBL 1 - FASE 3
GRUPO 3

Alexandre Miguel Rodrigues Nunes Pereira - 16/0000840
Daniel Maike Mendes Gonçalves - 16/0117003
Marco Antonio de Lima Costa - 16/0135681
João Vitor Ferreira Alves - 16/0127912
Pedro Rodrigues Pereira - 17/0062686
Renan Welz Schadt - 16/0143403
Rômulo Vinícius de Souza - 15/0147601
Shayane Marques Alcântara - 16/0144949

Brasília - DF

TBL 1 - FASE 3

GRUPO 3

Tabela de Contribuição

Integrantes	**Contribuição**
Alexandre Miguel Rodrigues Nunes Pereira	100
Daniel Maike Mendes Gonçalves	100
Marco Antonio de Lima Costa	100
João Vitor Ferreira Alves	100
Pedro Rodrigues Pereira	100
Renan Welz Schadt	100
Rômulo Vinícius de Souza	100
Shayane Marques Alcântara	100

SUMÁRIO

1. Walkthrough

É um método de teste realizado por meio de humanos, ou seja, não automatizado. Seu processo consiste em um time (idealmente até 4 pessoas) inspecionando o programa alvo durante uma reunião. A maioria das pessoas do time não são os autores do programa, indo em direção à questão de que não é eficiente o autor testar sua própria aplicação. Além disso, o processo pode vir a diminuir o custo de correção de erros, já que ele normalmente possibilita a localização de erros precisamente, economizando tempo.

Alguns estudos indicam que o uso das técnicas executadas por humanos mostraram-se mais efetivas do que processos automatizados, relação à qualidade dos testes. Com o Walkthrough, o processo de testes baseia-se na divisão de tarefas entre os membros da equipe, aliados à criação de casos de teste com as entradas e saídas esperadas, facilitando na organização e servindo para guiar o processo de teste. Os casos de teste podem ser usados também para questionar o programador em relação à lógica do programa, em que a maioria dos erros podem ser encontrados questionando.

A técnica é interessante para verificar se o código funcionará corretamente porque proporciona uma dinâmica acessível, com um detalhamento organizado e específico sobre a aplicação. Os questionamentos também para o autor do código é de extrema utilidade, visto que permite a identificação de erros mais rapidamente, tornando o processo mais eficiente. Além disso proporciona também a localização precisa de onde o erro está alocado, ocasionando no melhor aproveitamento do tempo.

2. Pseudocódigo

- SpaceX-API/spaceX.py

```
▷ Menu to choice the consult.
NEXT_LAUNCH ← 1
LAST_LAUNCH ← 2
UPCOMING_LAUNCHES ← 3
PAST_LAUNCHES ← 4

RUN(cls)
    ▷ Shows the options for the user
    ▷ :return: chosen option
    while true
        do
            print 'O que você deseja visualizar?'
            print '1) Próximo Lançamento'      ▷ next_launch
            print '2) Último Lançamento'       ▷ latest launch
            print '3) Próximos Lançamentos'    ▷ upcoming launches
            print '4) Lançamentos Passados'    ▷ past launches
            print '5) Sair'

            try
                print 'Insira uma opção: '
                read x
                option ← x
            except x not integer
                then
                    print 'Você deve inserir somente números inteiros de preferencia de 1 a 5'
                    option ← 0

            if option < 1 or option > 5
                then
                    print 'Essa opção não existe, por favor insira uma opção válida'
                    CLEAN[cls](3)
                    ▷ Clean cls
            else if option == 5
                then
                    CLOSE[cls]
                    break
            else
                SHOW_RESULT[cls](option)
                print 'Deseja sair da aplicação? (S/N): '
                read text
                answer ← text

                if answer[0] == 's' or answer[0] == 'S'
                    then
                        CLOSE[cls]
                        break

                CLEAN[cls](1)

```

```
SHOW_RESULT(cls, option)
    ▷ Run a specific option inserted
    ▷ :param option: Option to visualize some datas
    if option == NEXT_LAUNCH[SpaceX]
        then
            NEXT_LAUNCH[cls]
    else if option == LATEST_LAUNCH[SpaceX]
        then
            LATEST_LAUNCH[cls]
    else if option == UPCOMING_LAUNCHES[SpaceX]
        then
            UPCOMING_LAUNCHES[cls]
    else if option == PAST_LAUNCHES[SpaceX]
        then
            PAST_LAUNCHES[cls]
    else
        then
            print 'Opção invalida'

CLEAN(seconds)
    ▷ Clean the prompt
    SLEEP[time](seconds)

    if 'win' in platform[sys]
        then
            SYSTEM[os]('cls')
    else
        SYSTEM[os]('clear')

CLOSE()
    ▷ Close the program.

    print 'Finalizando o programa...'
    SLEEP[time](1)

NEXT_LAUNCH()
    ▷ Returns the next launch

    connection ← Connect('https://api.spacexdata.com/v3/launches/next')
    ▷ result for the next launch available

    print result[connection]

UPCOMING_LAUNCHES()
    ▷ Returns the closest upcoming launches

    connection ← Connect('https://api.spacexdata.com/v3/launches/upcoming')
    ▷ api result upcoming launch

    for result in result[connection]
        do
            print result
            print '------------------------------------------------------------'
```

```
104 LATEST_LAUNCH()
105     ▷ Returns the latest launch
106
107     connection ← Connect('https://api.spacexdata.com/v3/launches/latest')
108     ▷ api result for the last flights
109
110     print result[connection]
111
```

```
112 PAST_LAUNCHES()
113     ▷ Returns the past launches
114
115     connection ← Connect('https://api.spacexdata.com/v3/launches/past')
116     ▷ api result for the past flights
117
118     for result in result[connection]
119         do
120             print result
121             print '------------------------------------------------------------'
122
```

- SpaceX-API/models/launch.py

```
1 INIT(flight_number, mission_name, rocket, rocket_type, launch_success, launch_year, launch_date)
2     flight_number[Launch] ← flight_number
3     mission_name[Launch] ← mission_name
4     launch_date[Launch] ← launch_date
5     launch_year[Launch] ← launch_year
6     rocket[Launch] ← rocket
7     rocket_type[Launch] ← rocket_type
8     launch_success[Launch] ← launch_success
9
```

```
10 STR()
11     ▷ Give a object a string representation
12
13     if launch_success[Launch] is not null
14         then
15             return '((flight_number[Launch]) (mission_name[Launch]) (rocket[Launch])
16                     (launch_year[Launch]) (launch_date[Launch]) (launch_success[Launch])'
17
18     return '((flight_number[Launch]) (mission_name[Launch]) (rocket[Launch])
19             (launch_year[Launch]) (launch_date[Launch])'
20
```

```
21 FLIGHT_NUMBER()
22     return 'Número do Voo: (flight_number[Launch])'
23     ▷ return flight number
24
```

```
25 MISSION_NAME()
26     return 'Missão: (mission_name[Launch])'
27     ▷ return mission number
28
```

```
29 LAUNCH_DATE()
30     return 'Data de Lançamento (UTC): (launch_date[Launch]) às (launch_time[Launch])'
31     ▷ return launch date in format day/ month/year and launch_time in format hour/minute
32
```

```
33 LAUNCH_YEAR()
34     return 'Ano de Lançamento: (launch_year[Launch])'
35     ▷ return launch year
36
```

```
ROCKET()
    return 'Foguete: (rocket[Launch]) (rocket_type[Launch])'
    ▷ return rocket and rocket type

LAUNCH_SUCCESS()
    if launch_success[Launch]
        then
            return 'Lançamento realizado com sucesso!'
            ▷ return launch success

    return 'Lançamento falhou!'
    ▷ return launch failure
```

- SpaceX-API/connections/api_connection.py

```
CONNECT
    INIT(url, headers, params)
        ▷ Connection constructor to connect with API
        if headers not null
            ▷ Seed to build headers if value not informed
            then
                headers[Connet] ← headers
            else
                headers[Connet] ← {'Accept': 'application/json'}
        try
            if params not null
                then
                    ▷ Calling 'get' function on object 'request' imported from package,
                    ▷ using informed params as in requests.get(url, headers, params)
                    response[Connect] ← GET[requests](url, headers[Connect], params)

            else
                ▷ Calling function without informed params
                response[Connect] ← GET[requests](url, headers[Connect])

            ▷ Creating attribute 'result',
            ▷ obtained from the json returning fuction 'json()'
            ▷ from attribute 'response' from object 'Connect'
            result[Connect] ← json[response[Connect]]

        ▷ failed to connect with API through request
        except requests not valid
            print "Ocorreu um erro na comunicação com a API SpaceX"
```

```
    RESPONSE()
        ▷ Return response parameter from Connect
        return response[Connect]
```

```
RESULT()
    ▷ Get a JSON result of request and return it
    if result[Connect] is of dictionary type
        ▷ Build Launch object with parameters obtainder from get functions
        then
            ▷ Assign attribute result form class Connet
            result ← result[Connect]

            ▷ Get attributes from result parameter to build Launch class
            flight_number ← GET[result]('flight_number')
            mission_name ← GET[result]('mission_name')
            aux_rocket ← GET[result]('rocket')
            rocket ← GET[aux_rocket]('rocket_name')
            rocket_type ← GET[rocket]('rocket_type')
            launch_success ← GET[result]('launch_success')
            launch_date ← GET[result]('launch_date_utc')
            launch_year ← GET[result]('launch_year')

            ▷ Create object of Launch type from parameters obtained and return it
            launch ← Launch(flight_number,
                            mission_name,
                            rocket,
                            rocket_type,
                            launch_success,
                            launch_date,
                            launch_year)
        return launch

    ▷ Create empty list to latter return of launch objects
    launchs ← []

    ▷ For loop over elements in result parameter
    for i ← 0 to lenght[ result[Connect] ]
        ▷ Assign attribute result form class Connet
        result ← result[Connect]

        ▷ Get attributes from result parameter to build Launch class
        flight_number[Launch] ← GET[result]('flight_number')
        mission_name[Launch] ← GET[result]('mission_name')
        rocket ← GET[result]('rocket')
        rocket[Launch] ← GET[rocket]('rocket_name')
        rocket_type[Launch] ← GET[rocket]('rocket_type')
        launch_success[Launch] ← GET[result]('launch_success')
        launch_date[Launch] ← GET[result]('launch_date_utc')
        launch_year[Launch] ← GET[result]('launch_year')

        ▷ Build Launch object with parameters and add object to list
        launchs[i] ← Launch(flight_number,
                            mission_name,
                            rocket,
                            rocket_type,
                            launch_success,
                            launch_date,
                            launch_year)
    return launchs
```

3. Instância do Pseudocódigo

- Option == 1

```
main.py
  MAIN( )
    RUN( )
```

```
spaceX.py
  NEXT_LAUNCH ← 1
  LAST_LAUNCH ← 2
  UPCOMING_LAUNCHES ← 3
  PAST_LAUNCHES ← 4

  RUN(cls)
    while true
      do
        print 'O que você deseja visualizar?'
        print '1) Próximo Lançamento'
        print '2) Último Lançamento'
        print '3) Próximos Lançamentos'
        print '4) Lançamentos Passados'
        print '5) Sair'

        print 'Insira uma opção: '
        read 1
        option ← 1

        SHOW_RESULT[cls](1)
        print 'Deseja sair da aplicação? (S/N): '
        read 'S'
        answer ← 'S'

        if answer[0] == 's' or answer[0] == 'S'
          then
              CLOSE[cls]
              break

              CLEAN[cls](1)

  SHOW_RESULT(cls, 1)
    if 1 == NEXT_LAUNCH[SpaceX]
        then
            NEXT_LAUNCH[cls]

  NEXT_LAUNCH( )
    connection ← Connect('https://api.spacexdata.com/v3/launches/next')

    print result[connection]
```

```
  CLEAN(3)
        SLEEP[time](3)
        if 'win' in platform[sys]
            then
                SYSTEM[os]('cls')
        else
            SYSTEM[os]('clear')

  CLOSE( )
      print 'Finalizando o programa...'
      SLEEP[time](1)
```

```
api_connection.py
    INIT(url='https://api.spacexdata.com/v3/launches/next', headers=null, params=null)
      if headers not null → False
          then
              Connect.headers ← headers
          else
              Connect.headers ← {'Accept': 'application/json'}
      if params not null → False
          then
              response[Connect] ← GET[requests](url, headers[Connect], params)
          else
              response[Connect] ← GET[requests](url, headers[Connect])

          result[Connect] ← json[response[Connect]]

          Connect {
            url ← 'https://api.spacexdata.com/v3/launches/next'
            headers ← {'Accept': 'application/json'}
            params ← null
            result ← {'flight_number': 84, 'mission_name': 'Starlink 2', 'mission_id': [],
 'launch_year': '2019', 'launch_date_unix': 1571270400, 'launch_date_utc': '2019-10-17T00:00:00.000Z',
 'launch_date_local': '2019-10-16T20:00:00-04:00', 'is_tentative': True, 'tentative_max_precision':
 'day', 'tbd': False, 'launch_window': None, 'rocket': {'rocket_id': 'falcon9', 'rocket_name': 'Falcon
9', 'rocket_type': 'FT', 'first_stage': {'cores': [{'core_serial': None, 'flight': None, 'block': 5,
 'gridfins': True, 'legs': True, 'reused': True, 'land_success': None, 'landing_intent': True,
 'landing_type': 'ASDS', 'landing_vehicle': 'OCISLY'}]}, 'second_stage': {'block': 5, 'payloads':
 [{'payload_id': 'Starlink 2', 'norad_id': [], 'reused': False, 'customers': ['SpaceX'], 'nationality':
'United States', 'manufacturer': 'SpaceX', 'payload_type': 'Satellite', 'payload_mass_kg': None,
 'payload_mass_lbs': None, 'orbit': 'VLEO', 'orbit_params': {'reference_system': 'geocentric',
 'regime': 'very-low-earth', 'longitude': None, 'semi_major_axis_km': None, 'eccentricity': None,
 'periapsis_km': None, 'apoapsis_km': None, 'inclination_deg': None, 'period_min': None,
 'lifespan_years': None, 'epoch': None, 'mean_motion': None, 'raan': None, 'arg_of_pericenter': None,
 'mean_anomaly': None}}]}, 'fairings': {'reused': False, 'recovery_attempt': False, 'recovered': False,
'ship': None}}, 'ships': [], 'telemetry': {'flight_club': None}, 'launch_site': {'site_id':
 'ccafs_slc_40', 'site_name': 'CCAFS SLC 40', 'site_name_long': 'Cape Canaveral Air Force Station Space
Launch Complex 40'}, 'launch_success': None, 'links': {'mission_patch': None, 'mission_patch_small':
 None, 'reddit_campaign': None, 'reddit_launch': None, 'reddit_recovery': None, 'reddit_media': None,
 'presskit': None, 'article_link': None, 'wikipedia': None, 'video_link': None, 'youtube_id': None,
 'flickr_images': []}, 'details': 'This mission will launch the first batch of Starlink version 1.0
satellites, from SLC-40, Cape Canaveral AFS. They are expected to contribute to the 550 km x 53° shell.
It is the second Starlink launch overall. Starlink is a low Earth orbit broadband internet
constellation developed and owned by SpaceX which will eventually consist of nearly 12 000 satellites
and will provide low latency internet service to ground terminals around the world. The booster for
this mission is expected to land on OCISLY.', 'upcoming': True, 'static_fire_date_utc': None,
 'static_fire_date_unix': None, 'timeline': None, 'crew': None}
            }
```

```
    RESPONSE()
        return response[Connect]

    RESULT()
        if result[Connect] is of dictionary type
            then
                result ← result[Connect]

                flight_number ← GET[result]('flight_number')
                mission_name ← GET[result]('mission_name')
                aux_rocket ← GET[result]('rocket')
                rocket ← GET[aux_rocket]('rocket_name')
                rocket_type ← GET[rocket]('rocket_type')
                launch_success ← GET[result]('launch_success')
                launch_date ← GET[result]('launch_date_utc')
                launch_year ← GET[result]('launch_year')

                launch ← Launch(flight_number, mission_name, rocket, rocket_type, launch_success,
launch_date, launch_year)
            return launch
```

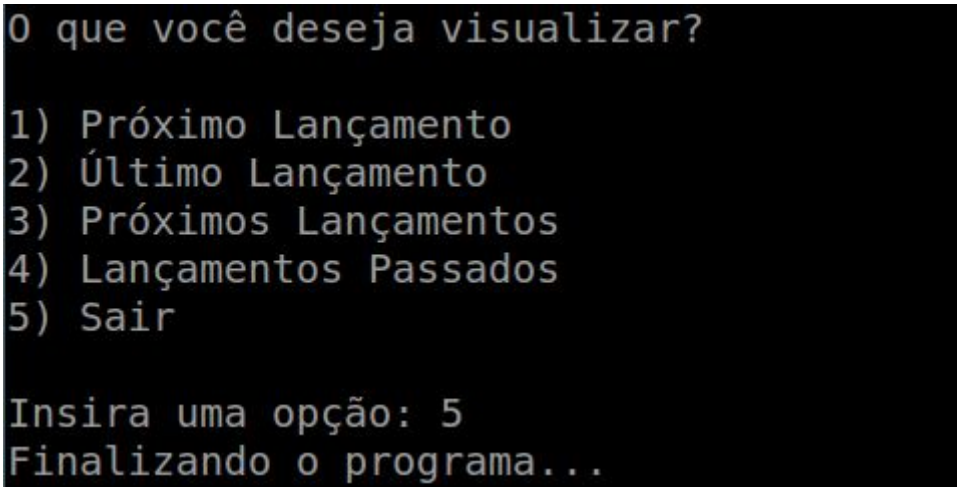
launch.py

```
    INIT(flight_number, mission_name, rocket, rocket_type, launch_success, launch_year, launch_date)
        flight_number[Launch] ← flight_number
        mission_name[Launch] ← mission_name
        launch_date[Launch] ← launch_date
        launch_year[Launch] ← launch_year
        rocket[Launch] ← rocket
        rocket_type[Launch] ← rocket_type
        launch_success[Launch] ← launch_success

    FLIGHT_NUMBER()
        return 'Número do Voo: 84'

    MISSION_NAME()
        return 'Missão: Starlink 2'

    LAUNCH_DATE()
        return 'Data de Lançamento (UTC): 17/10/2019 às 00:00'

    LAUNCH_YEAR()
        return 'Ano de Lançamento: 2019'

    ROCKET()
        return 'Foguete: Falcon9 (FT)'

    LAUNCH_SUCCESS()
        if launch_success[Launch]
            then
                return 'Lançamento realizado com sucesso!'
        return 'Lançamento falhou!'
```

```
O que você deseja visualizar?

1) Próximo Lançamento
2) Último Lançamento
3) Próximos Lançamentos
4) Lançamentos Passados
5) Sair

Insira uma opção: 1

Número do Voo: 84
Missão: Starlink 2
Foguete: Falcon 9 (FT)
Ano de Lançamento: 2019
Data de Lançamento (UTC): 17/10/2019 às 00:00

Deseja sair da aplicação? (S/N): s
Finalizando o programa...
```

- Option == 5

```
main.py
  MAIN( )
    RUN( )

spaceX.py
  NEXT_LAUNCH ← 1
  LAST_LAUNCH ← 2
  UPCOMING_LAUNCHES ← 3
  PAST_LAUNCHES ← 4

  RUN(cls)
    while true
      do
        print 'O que você deseja visualizar?'
        print '1) Próximo Lançamento'
        print '2) Último Lançamento'
        print '3) Próximos Lançamentos'
        print '4) Lançamentos Passados'
        print '5) Sair'

        print 'Insira uma opção: '
        read 5
        option ← 5

        if option < 1 or option > 5
            then
                print 'Essa opção não existe, por favor insira uma opção válida'
                CLEAN[cls](3)
                ▷ Clean cls
        else if option == 5
            then
                CLOSE[cls]
                break
  CLEAN(3)
        SLEEP[time](3)
        if 'win' in platform[sys]
            then
                SYSTEM[os]('cls')
        else
            SYSTEM[os]('clear')

  CLOSE( )
      print 'Finalizando o programa...'
      SLEEP[time](1)
```

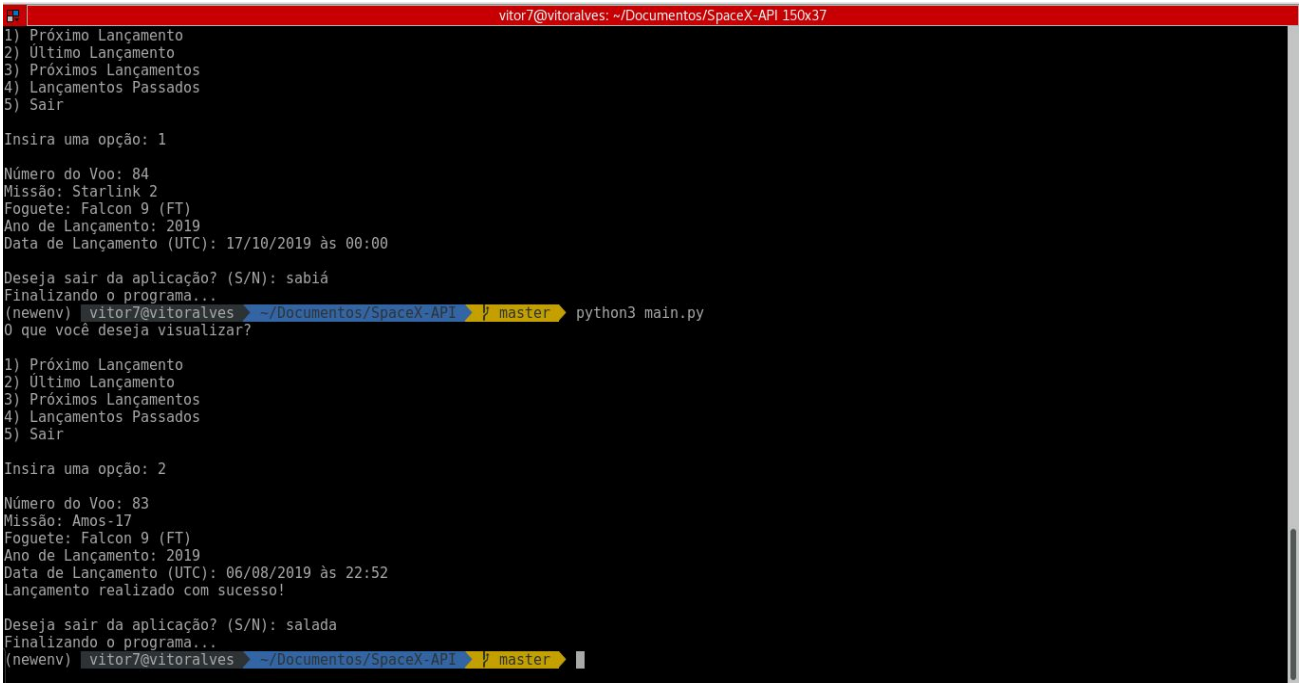


4. Código

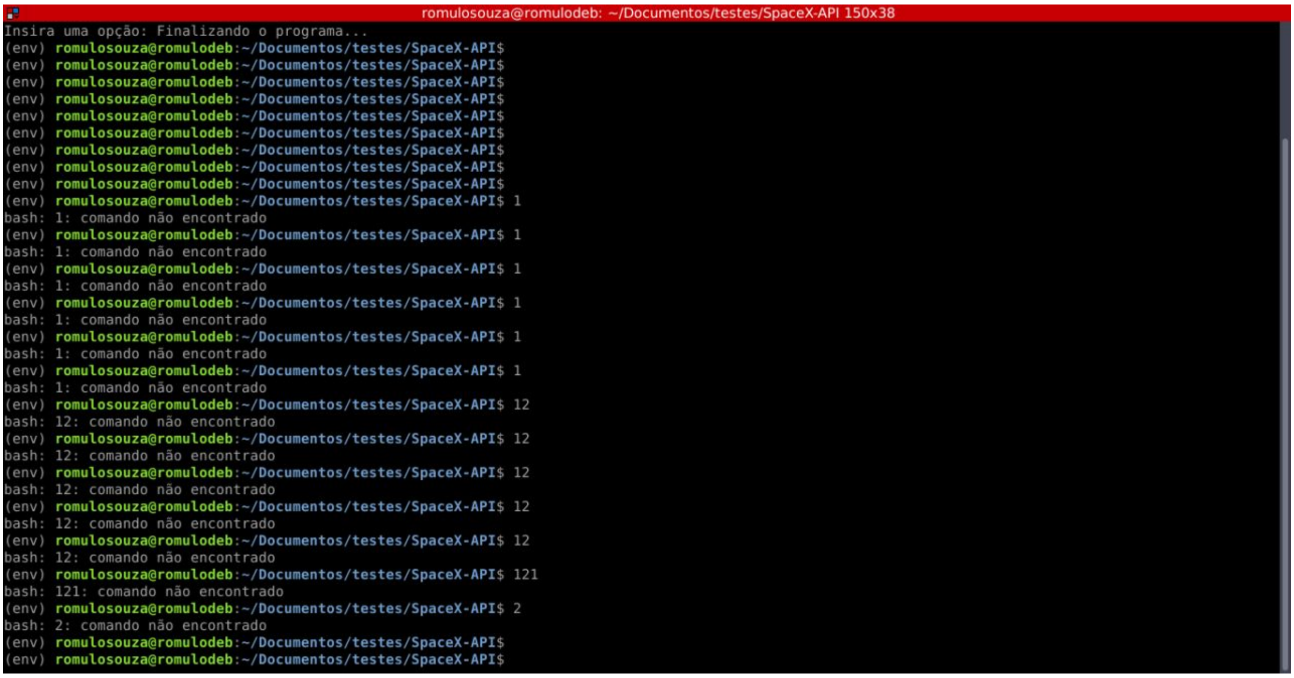
4.1. Análise do Código

Em busca de problemas e erros no código, chegamos a alguns comportamentos anômalos, os quais são:

- Na opção de escolha para sair do programa após a realização de uma atividade é possível inserir qualquer palavra que comece com 'S' ou 's' e o programa se encerra, sem validar se corresponde a apenas o ‘S’ ou ‘s’. Essa validação, no código, é feita verificando o primeiro caractere de cada palavra. Para solucionar esse erro, poderia ser feito uma validação da palavra como um todo: transformar a palavra totalmente para minúsculo e, então, verificar se ela corresponde a “sim”, “s” e, nas demais opções, inferir que foi uma escolha negativa.



- A cada opção selecionada existe um tempo de espera para que seja selecionada outra opção do menu. Quando várias opções são selecionadas em sequência, antes que o tempo de espera tenha terminado, o software não apresenta o comportamento esperado. Em um dos testes realizados, após se passar a opção de finalizar o programa (quando o mesmo estava em estado de espera de opções anteriores), o programa continuou aceitando entradas e, depois que a fila de entradas chegou na opção que finaliza o programa, as entradas que vieram após foram inseridas diretamente no bash. Para resolver esse problema, pode-se remover o timeout que é dado a cada comando.



4.2. Qualidade de Código

Quanto a análise estática do código, também foram encontrados defeitos. Uma boa prática, quando se desenvolve em python, é seguir o guia de estilo fornecido pelo pep8. No pep8, são padronizados espaços, tamanho de linhas, indentação, imports, dentre outros. No código em questão, ao se analisar se está de acordo com o pep8, pode-se observar erros como:

- Múltiplos imports em uma mesma linha
- Linhas com mais de 79 caracteres
- Múltiplas linhas em branco
- Falta de linha em branco no fim do arquivo
- Imports não utilizados

Entende-se como qualidade de software uma gama de práticas e métricas a serem seguidas, as quais demandam tempo e recursos, mas retornam benefícios ao produto final. Com uma boa qualidade de código é possível tornar maior a manutenibilidade de um produto, assim reduzindo seus custos de manutenção. Com uma boa gerência e processos de trabalho bem definidos, observa-se um ganho considerável de qualidade ao software.

Dessa forma, podemos definir qualidade como tudo aquilo que indica, em fatores contábeis, a produção e produto final de um software. Além disso, possui pontos de visão e fatores diferentes, que se enquadrarão em cada contexto necessário.

5. Evidências de Execução

Com o objetivo de demonstrar o funcionamento da aplicação foram coletadas as evidências de execução. A seguir demonstra-se as telas de execução do sistema.

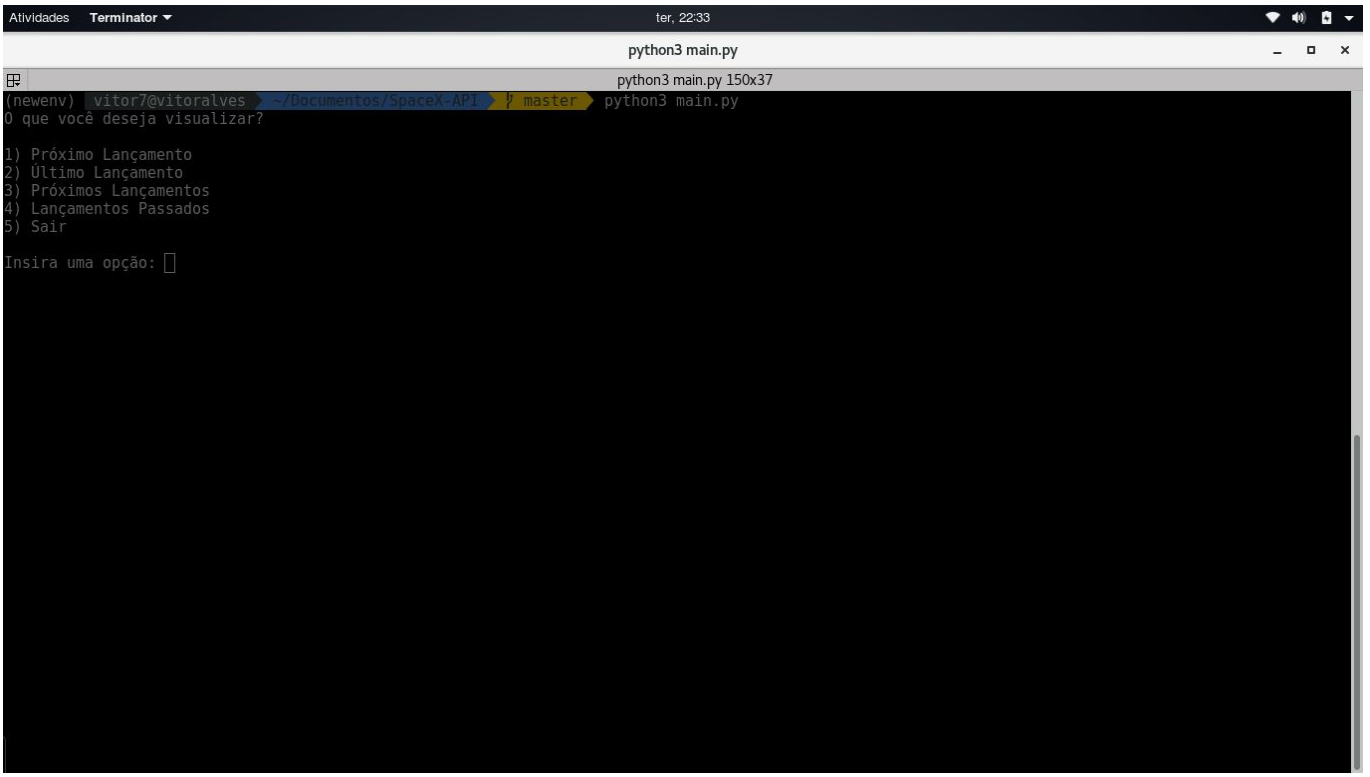


Figura 5.1 – Tela de Menu principal

A tela acima é a primeira tela de execução do sistema, utilizada para navegar em sua funcionalidades. Após isso, foi selecionado a primeira opção do menu.

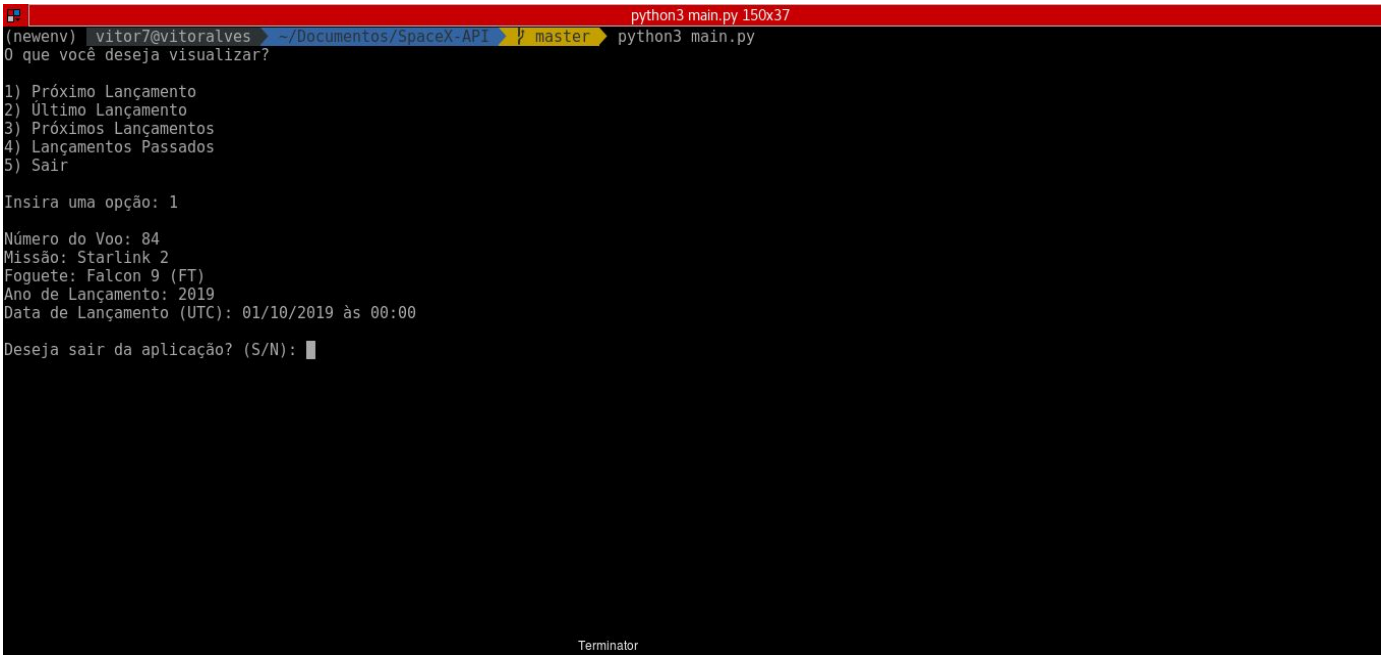


Figura 5.2 – Tela de Próximo Lançamento

Na imagem 5.2 vemos as informações sobre o próximo lançamento como o número do Voo, a missão, foguete, Ano e data do lançamento. Após isso selecionando a opção N ela limpa o terminal e nos retorna novamente o menu.

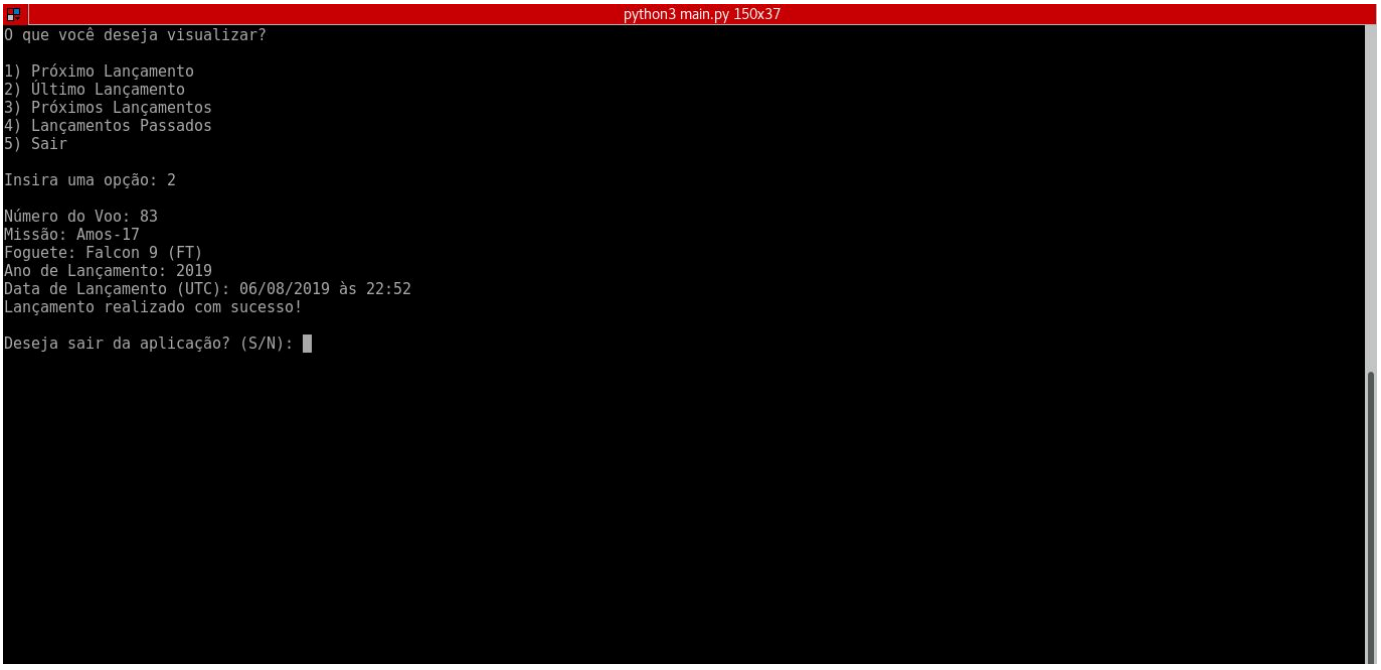


Figura 5.3 – Tela de Último lançamento

Após selecionar a opção 2 do menu ela nos retorna as informações do último lançamento como número do Voo, a missão, foguete, Ano, data do lançamento e se o lançamento foi efetuado com sucesso. Depois selecionamos a opção N e retomamos ao menu inicial.

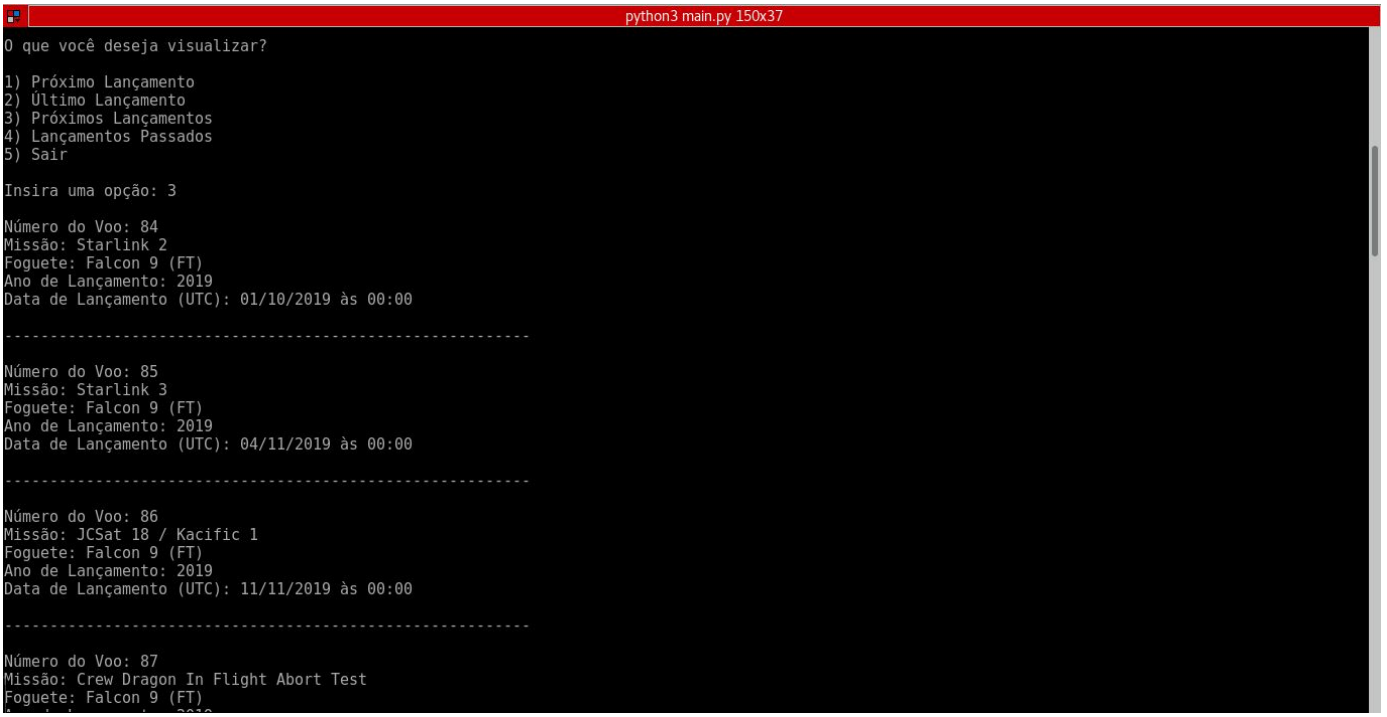


Figura 5.4.1 – Tela de Próximos Lançamentos

Depois de selecionar a opção 3 do menu o programa nos apresenta quais serão os próximos lançamentos que irão acontecer. Com a informações sobre número do Voo, missão, foguete, Ano e a data do lançamento de cada um dos lançamentos.

Figura 5.4.2 – Tela de Próximos Lançamentos

Após isso seleciona-se a opção N para não sair da aplicação. Dessa forma, retorna-se ao menu iniciar.

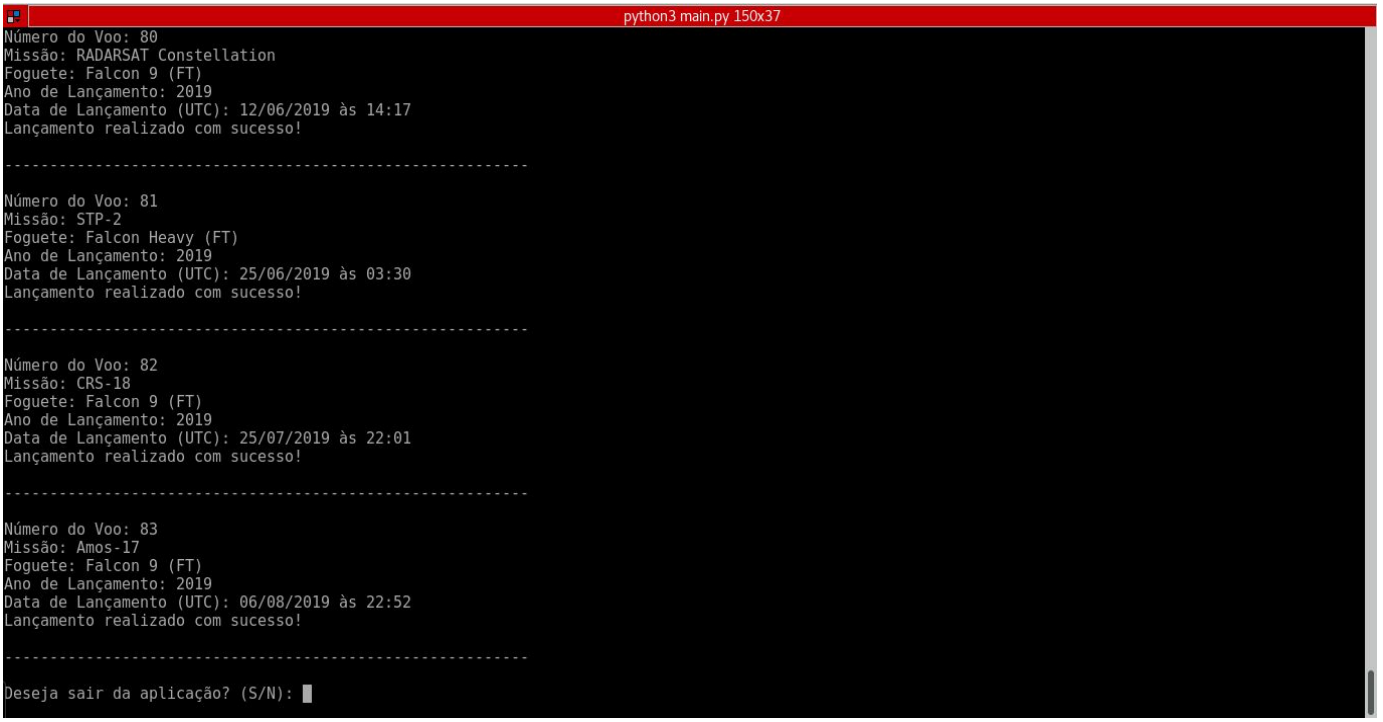


Figura 5.5 – Tela de Lançamentos Passados

Selecionando a opção 4 de lançamentos passados é apresentado todos os lançamentos feitos pela SpaceX. Mostra as informações sobre cada lançamento, como o número do Voo, a missão, foguete, Ano, data do lançamento e se o lançamento foi efetuado com sucesso. Após isso selecionando a opção N voltamos ao menu inicial.

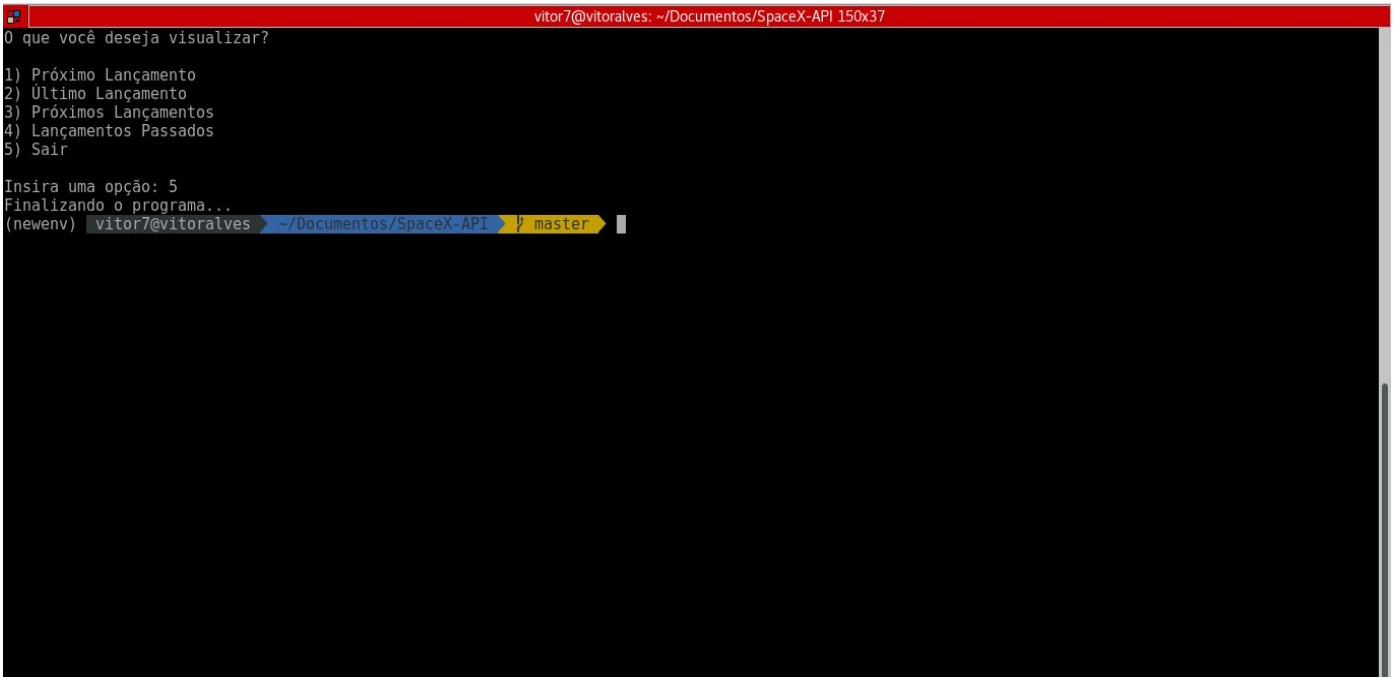


Figura 5.6 – Tela de Sair do Sistema

Por fim, seleciona-se a opção 5 para sair do sistema.

REFERÊNCIAS

CORMEN, T., LISERSON, C., RIVEST, R., et al., 2002, **Algoritmos Teoria e prática**. . 2nd Ed. S.l., Elsevier.

MEYERS, G., BADGETT, T., SANDLER, C., et al., 2004, **The Art of Software Testing**. . 2nd Ed. JW&S., Inc.