СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc32334397)

[1 ОПИСАНИЕ ОРГАНИЗАЦИИ (ОРГАНИЗАЦИИ) 4](#_Toc32334398)

[1.1 Краткая характеристика организации 4](#_Toc32334399)

[1.2 Структура организации 4](#_Toc32334400)

[2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ ДЛЯ ОРГАНИЗАЦИИ ПРОИЗВОДСТВЕННОГО ПРОЦЕССА 6](#_Toc32334401)

[2.1 Системное программирование 6](#_Toc32334402)

[2.1.1 Оснащённость организации системным программным обеспечением 6](#_Toc32334403)

[2.1.2 Характеристики системного программного обеспечения 6](#_Toc32334404)

[2.2 Прикладное программирование 7](#_Toc32334405)

[2.2.1 Оснащённость организации прикладным программным обеспечением 7](#_Toc32334406)

[2.2.2 Характеристики прикладного программного обеспечения 7](#_Toc32334407)

[2.2.3 Техническое задание решаемой задачи 9](#_Toc32334408)

[2.2.4 Описание алгоритмов реализации модулей 10](#_Toc32334409)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 13](#_Toc32334410)

[ПРИЛОЖЕНИЕ А 14](#_Toc32334411)

[ПРИЛОЖЕНИЕ B 20](#_Toc32334412)

[ПРИЛОЖЕНИЕ С 21](#_Toc32334413)

# ВВЕДЕНИЕ

В большинстве учебных заведениях есть компьюторные классы, в которых преподоют информатику. И для того, чтобы учащиеся получили задания учитель или преподователь должен отправить на каждый компьютор фаил с заданием через Totol cammander или припомощи проводника, при этом тратится кучу времени на отправку 1 файла, а ведь копмьютеров в компьюторном классе больше 10, и чтобы раскидать 1 файл навсе компьютеры требуется большое количесво времени.

Програм в свободном доступе которые, которые могут отправить на 1 компьютор файл очень мало, и они имеють большинство ненужных функций либо же они платнные, а что ещё уже они могудт доваться лишь на короткий периуд.

Поэтому была создана программа «AMFOTLN», которая позволяет раскидать большое или малое количество файлов по компьюторам связаныйх в 1 локальную сеть.

# 1 ОПИСАНИЕ ОРГАНИЗАЦИИ (ОРГАНИЗАЦИИ)

1.1 Краткая характеристика организации

Организация является федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский ядерный университет "МИФИ" в лице и.о. руководителя Новоуральского технологического института - филиала федерального государственного автономного образовательного учреждения высшего образования "Национальный исследовательский ядерный университет "МИФИ" Зиновьева Григория Сергеевича.

После опроса сотрудников и учащихся была выевлина жалоба что перподователи долго раскидывают файлы на все компьютеры что сокрощает время учебного процесса.

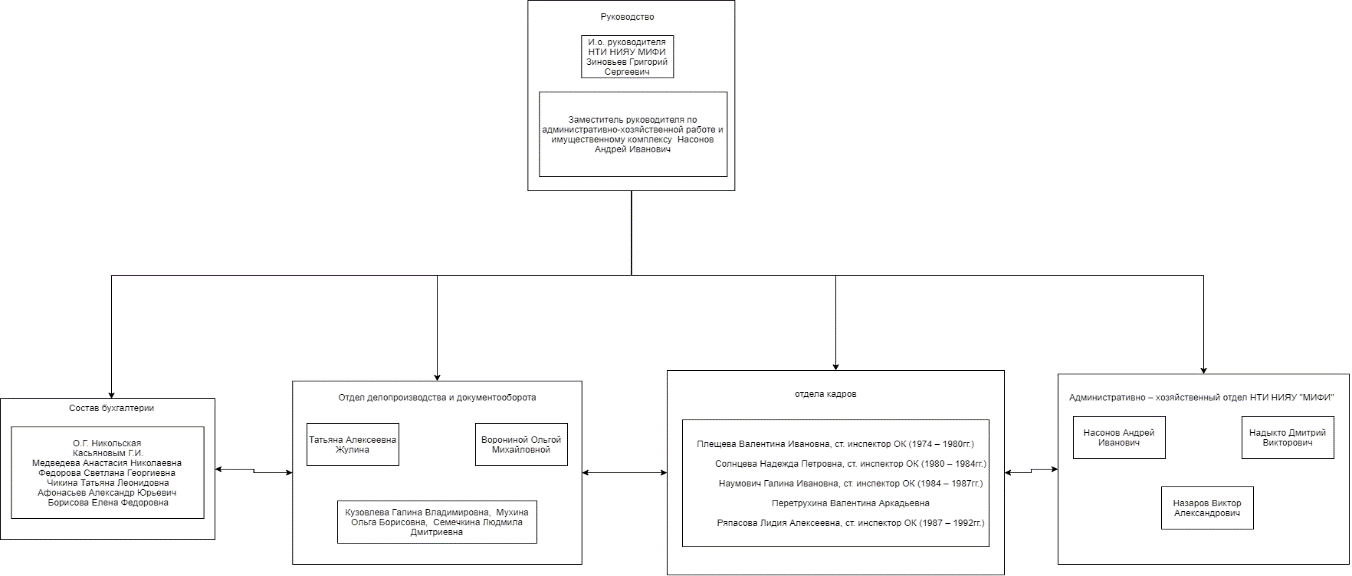
Таким образом целью работы является разработка приложения «AMFOTLN» для быстрого распределения файлов в локальной сети.

Было принято решение о создании приложения по причине того, что на организация есть люди недовольные тем что преподователи долго раскидывают файлы по компьютерам.

Данное приложение сможет помочь организации тем, что оно самостоятельно отправит паролельно вайл или папку на неогрониченное количество серверов.

1.2 Структура организации

Структура организации НТИ НИЯУ МИФИ представлена на рисунке «Структура организации»

Рисунок 1 – «Структура организации»

ВУЗ это сложная организм. Это не только преподаватели и студенты. это администрация и учебный отдел, библиотека и спецчасть, делопроизводство и бухгалтерия, отдел кадров и хозяйственная служба и мног кто ещё. В схемах невозможно точно отоброзить как работоет столь сложный организм веди все другдругу помогают и поодерживают, помогают и будем честные умалчиваут некоторые промашки коллег.

# 2 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, ИСПОЛЬЗУЕМОЕ ДЛЯ ОРГАНИЗАЦИИ ПРОИЗВОДСТВЕННОГО ПРОЦЕССА

2.1 Системное программирование

2.1.1 Оснащённость организации системным программным обеспечением

На организации установлена 2 основные системы где первой системой является Windows, причём она начинается от Windows xp и заанчивается Windows 10, а второй системой является Linux он держдит все сервера и даёт возможность обмениваться файлами во внутреней сети. Основными серверами является сервер MAIN он же 192.168.0.2 на которм идёт обмен файлами между преподователями и при этом которые не надо постоянно хранить поскольку его котологи очищаются раз в неделю, и есть ещё 192.168.0.3 на котором уже хранятся файлы на постоянной основе и в гораздо больших обёмов, и ещё есть сервер который раздоёт всем интрнэт его адресом является 192.168.0.3.

2.1.2 Характеристики системного программного обеспечения

Операционная система macOS Mojave поддерживается моделями MacBook Air серидины 2012 года и выше, iMac, Mac Mini конца 2012 года и новее, а также Mac Pro конца 2013, серидины 2010 и серидины 2012 года. [3]

Системные требования macOS Mojave предоставлены ниже [3]:

* OS X Mountain Lion 10.8 или выше
* 4 Гб RAM
* 15-20 Гб свободного места на диске

2.2 Прикладное программирование

2.2.1 Оснащённость организации прикладным программным обеспечением

В качестве прикладного программного обеспечения на организации используется Microsoft® World 2001-20013.

В качестве браузеров используются:

* Google Chrome
* Firefox
* Яндекс.Браузер Версия

В качестве IDE используются Intellij IDE, а в качестве редакторjd используется Word, Excel.

2.2.2 Характеристики прикладного программного обеспечения

Microsoft® World - это самое совершенное решение для работы с текстовыми документами. Узнайте больше о средствах совместного редактирования в Word. ... С этими удобными инструментами вы запросто и без всяких инструкций сможете перейти от ручки и бумаги к цифровому рукописному вводу и редактированию. Сохраняйте концентрацию. Получайте всю необходимую информацию прямо в Word, не отвлекаясь от работы над документом.[1]

Google Chrome – это многофункциональный помощник, который востребован новичками, опытными пользователями и разработчиками. [2]

Яндекс.Браузер предоставляет возможность работать при медленном интернете, ведь если интернет медленный, включается режим Турбо — он ускоряет загрузку сайтов и видео. [3]

Intellij IDE — в первую очередь IDE для Java, она понимает и предоставляет интеллектуальную помощь при написании кода на SQL, JPQL, HTML, JavaScript и многих других языках и позволяет редактировать код, написанный не на Java, внутри строковых литералов Java-кода. Продуктивная работа. IntelliJ IDEA анализирует однообразные задачи в процессе разработки и автоматизирует их, чтобы вы могли сосредоточиться на общей картине. Эргономичная среда. [4]

Firefox — больше, чем просто браузер. Узнайте больше о продуктах Firefox, которые относятся к вашим данным с уважением и созданы для обеспечения приватности в любом месте сети.

2.2.3 Техническое задание решаемой задачи

Разрабатываемый программный продукт должен позволять выберать файл для отправки,сохранять данные введённые пользователем,выводить уведомления, а также выдовать количество завершонных процессов.

Техническое задание решаемой задачи представлено в приложении А.

2.2.4 Описание алгоритмов реализации модулей

Описание алгоритмов реализации модулей представлены на рисунке 2 – «Схема программы».

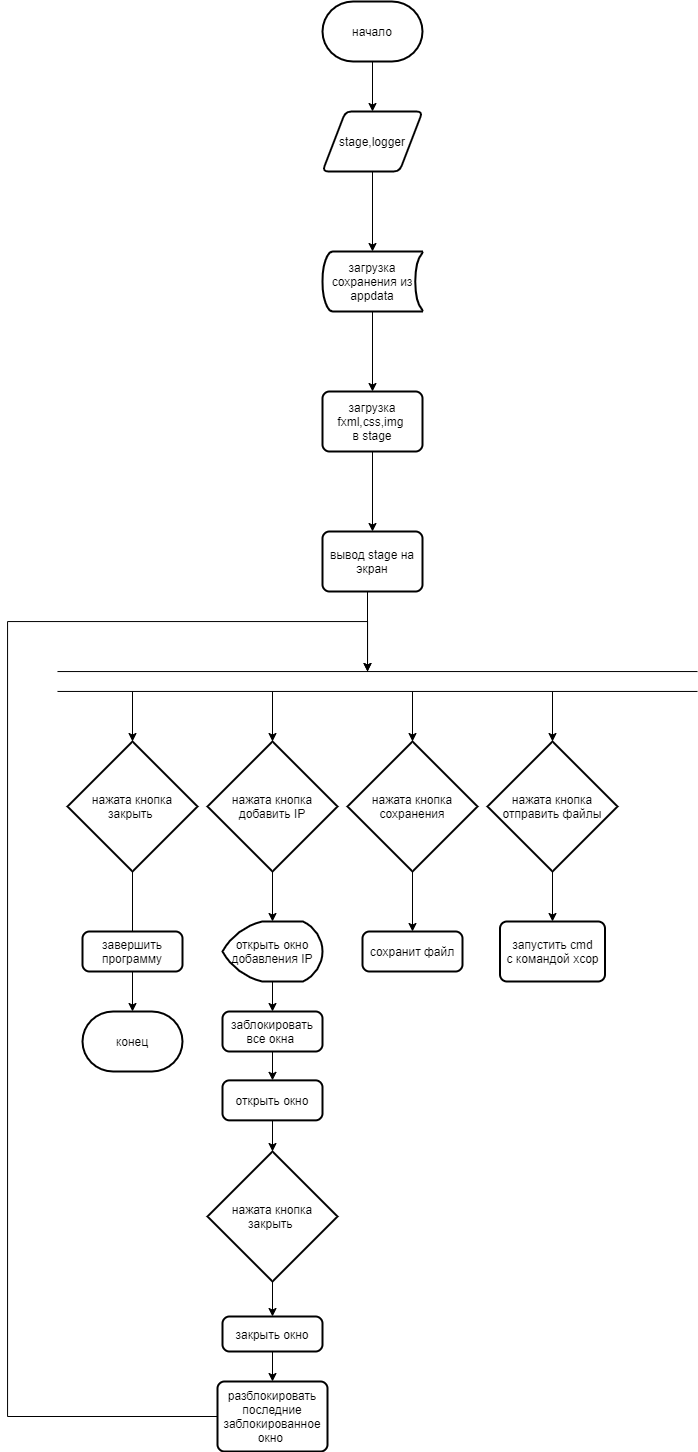


Рисунок 2 – «Схема программы»

**2.2.5 Тестирование и отладка программного продукта**

Для тестов использовался debag встроеный в IDEA, а также компьюторный классе под номером 107 в котором были раскиданны 2 файлв «Autocad 2020» и «windows 10 update». В калассе успешно были отправлены все файлы кроме одного по скольку он был в не сети поскольку он был загружен из тестового файла сохранения. Входе тестирования были обноружены ошибки связоные с невозможностью просмотреть файлы на удолённом компьюторе и получение их из cmd.

**2.2.6 Документирование программного продукта**

Документация к приложению «SwapiService» представлена в следующих документах:

* Техническое задание представлено в приложении А
* Описание программы представлено по тексту отвчёта
* Руководство пользования представлено в приложении В

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Microsoft Word [Электронный ресурс] – режим доступа: https://products.office.com/ru-ru/Word?rtc=1 (дата обращения 12.02.2020)
2. Google Chrome [Электронный ресурс] – режим доступа: https://googlechro-me.ru/chto-takoe-chrome/ (дата обращения 12.02.2020)
3. Яндекс Браузер [Электронный ресурс] – режим доступа: https://browser.yandex.ru/ (дата обращения 12.02.2020)
4. IntelliJ IDEA [Электронный ресурс] – режим доступа: https://www.jetbrains.com/ru-ru/idea/ (дата обращения 12.02.2020)
5. https://www.mozilla.org/ru/firefox/

# ПРИЛОЖЕНИЕ А

Техническое задание

**1 Введение**

**1.1 Наименование программы**

Наименование программы «SwapiService».

**1.2 Краткая характеристика области применения**

Программа предназначена для предоставления доступа к базе данных, хранящую информацию о персонажах, планетах и комсмических кораблях из саги "Звёздные войны".

Программа предоставляет веб-интерфейс в браузере для получения доступа к базе данных.

**2 Основания для разработки**

Основанием для разработки является индивидуальное задание на производственную практику по профессиональному модулую ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем, утверждённого начальником отдела практики и трудоустройства С.Л. Ждановским 14 января 2020 года.

**3 Назначение разработки**

Назначением разработки является повышение производительности сотрудников, за счёт добавления нового развлечения во время перерыва на работе.

**4 Требования к программе**

**4.1 Требования к функциональным характеристикам**

Программа должна предусматривать следующие возможности:

* Просмотр списка персонажей и их детельного представления по выбору на одной странице
* Просмотр списка планет и их детельного представления по выбору на одной странице
* Просмотр списка космических кораблей и их детельного представления по выбору на одной странице
* Просмотр на каждой странцие случайной планеты

**4.2 Требования к надежности**

Требованиями к надёжности являются:

* Скрытие от пользователя пугающих ошибок
* Добавление индикатора загрузки к динамически изменяющимся компонентам

**4.3 Условия эксплуатации**

В условия эксплуатация входит:

* Оплата сервера по подписке, купленного на Yandex Cloud.
* Оплата доменного имени на сервисе REG.RU
* Любой SSL сертификат

**4.4 Требования к составу и параметрам технических средств**

В качестве сервера рекомендуется использовать сервер со следующими характеристиками:

* Платформа - Intel Cascade Lake
* Гарантированная доля vCPU - 5%
* vCPU - 2
* RAM - 1 ГБ
* Объём дискового пространства - 5 ГБ

**4.5 Требования к информационной и программной совместимости**

В требования к информационной и программной совместимости входят:

* Сервер с установленной ОС Linux Debian
* На сервре должен быть установлен NodeJS, ReactJS, React Router.
* На сервере должен быть установлен Python 3.8, Django, Django REST Framework.
* На сервере должна быть устновлена PostgresQL

**4.6 Требования к маркировке и упаковке**

Требования к маркеровке не предусматриваются.

Требованием к упаковке является доступ к публичному репозиторию на GitHub.

**4.7 Требования к транспортированию и хранению**

Требования к транспортированию и хранению не предусматриваются.

**5 Требования к программной документации**

Программа должна содержать следующую прораммную документацию:

* Техническое задание
* Описание программы, представленной в виде отчёта по ПП.01.01
* Инструкция пользования

**6 Технико-экономические показатели**

Предполагается, что ориентировочная экономическая эффективность увеличится, за счёт увеличения мотивации к работе, от использования программы.

Экономическим преимуществом по сравению с лучшими отечественными и зарубежными образцами или аналогами является использование современных технологий разработки сайтов, называемых «Single Page Aplication».

**7 Стадии и этапы разработки**

Стадии и этапы разработки, а так же их сроки исполнения представлены в таблице 1 – «Стадии и этапы разработки»

Таблица 1 – «Стадии и этапы разрабокти»

|  |  |
| --- | --- |
| **Стадии и этапы** | **Сроки исполнения** |
| Ознакомление со структурой и характером деятельности организации. Ознакомление с программным обеспеченеим, используемым для организации производствнного процесса | 03.02.2020 |
| Подбор совместно с руководтелем производственной практики (на рабочем месте) задачи для разработки прораммного решения реализации одного из информационных процессов организации | 04.02.2020 |
| Составление технического задания для задачи, сформулированной руководителем производственной практики (на рабочем месте) | 05.02.2020 |
| Разработка, оформление алгоритма решения задачи поставленной задачи (на рабочем месте) | 06.02.2020 - 07.02.2020 |
| Разработка программного продукта | 08.02.2020 - 12.02.2020 |
| Отладка и тестирование программного продукта | 13.02.2020 |
| Внедрение программного продукта | 14.02.2020 |
| Оформление отчёта по ПП.01.01 | 12.02.2020 |

**8 Порядок контроля и приёмки**

На основании Протокола проведения испытаний Исполнитель совместно с Заказчиком подписывает Акт приемки-сдачи программы в эксплуатацию.

# ПРИЛОЖЕНИЕ B

Инструкция пользования

На всех страницах сайта сверху присутсвует навигационная панель. Для того, чтобы перейти к интерисующему разделу, необходимо нажвать на одну из предложенных кнопок:

* People
* Planets
* Starships

Для изменения сервера, с которого берутся данные для сайта, необходимо нажать на кнопку «Change Service».

Чтобы связаться с автором сайта, необходимо нажать на кнопку «Developer».

# ПРИЛОЖЕНИЕ С

Листинг программы

// index.js

import ReactDOM from "react-dom"

import React from "react"

import \* as serviceWorker from './serviceWorker'

import App from "./components/app"

ReactDOM.render(<App />, document.getElementById('root'))

serviceWorker.register()

// app.js

import React, { Component } from "react"

import Header from "../header"

import RandomPlanet from "../random-planet"

import SwapiService from "../../services/swapi-service"

import DjangoSwapiService from "../../services/django-swapi-service.js"

import ErrorBoundary from "../error-boundary"

import { SwapiServiceProvider } from "../swapi-service-context"

import {

PeoplePage,

PlanetsPage,

StarshipsPage,

WelcomePage,

LoginPage,

SecretPage

} from "../pages"

import { BrowserRouter as Router, Route, Switch, Redirect } from "react-router-dom"

export default class App extends Component {

state = {

swapiService: new DjangoSwapiService(),

isLoggedIn: false

}

onLogin = () => {

this.setState({

isLoggedIn: true

})

}

onServiceChange = () => {

this.setState(({ swapiService }) => {

const Service = swapiService instanceof SwapiService ? DjangoSwapiService : SwapiService

return {

swapiService: new Service()

}

})

}

render() {

const { isLoggedIn } = this.state

return (

<ErrorBoundary>

<SwapiServiceProvider value={this.state.swapiService}>

<Router>

<Header onServiceChange={this.onServiceChange} />

<Switch>

<Route path="/" render={() => <WelcomePage />} exact />

<Route path="/people/:id?" component={PeoplePage} />

<Route path="/planets/:id?" component={PlanetsPage} />

<Route path="/starships/:id?" component={StarshipsPage} />

<Redirect to="/" />

</Switch>

<RandomPlanet updateInterval={10000} />

</Router>

</SwapiServiceProvider>

</ErrorBoundary>

)

}

}

// people-page.js

import React, { useEffect } from "react"

import { PersonDetails, PersonList } from "../sw-components"

import { withRouter } from "react-router-dom"

import scrollToTop from "../scroller"

const useScroll = (value) => {

useEffect(() => {

scrollToTop()

}, [value])

}

const PeoplePage = ({ history, match }) => {

const { id } = match.params

useScroll(id)

return (

<React.Fragment>

<PersonDetails itemId={id} />

<PersonList

onItemSelected={(id) => history.push(id)} />

</React.Fragment>

)

}

export default withRouter(PeoplePage)

// person-details.js

import ItemDetail, { Record } from "../item-detail"

import React from "react"

import { withSwapiService } from "../hoc-helper"

const PersonDetails = (props) => {

return (

<ItemDetail {...props}>

<Record field="gender" label="Gender" />

<Record field="eyeColor" label="Eye color" />

<Record field="birthYear" label="Birth year" />

</ItemDetail>

)

}

const mapMethodsToProps = (swapiService) => {

return {

getData: swapiService.getPerson,

getImageUrl: swapiService.getPersonImage

}

}

export default withSwapiService(mapMethodsToProps)(PersonDetails)

// item-lists.js

import React from "react"

import BodySelect from "../body-select/body-select"

import { withData, withSwapiService, withChildFunction, compose } from "../hoc-helper"

const renderName = ({ name }) => <span>{name}</span>

const renderModelAndName = ({ model, name }) => <span>{name} [{model}]</span>

const mapPersonMethodsToProps = (swapiService) => {

return {

getData: swapiService.getAllPeople

}

}

const mapPlanetMethodsToProps = (swapiService) => {

return {

getData: swapiService.getALlPlanets

}

}

const mapStarshipMethodsToProps = (swapiService) => {

return {

getData: swapiService.getALlStarships

}

}

const PersonList =

compose(

withSwapiService(mapPersonMethodsToProps),

withData,

withChildFunction(renderName)

)(BodySelect)

const PlanetList =

compose(

withSwapiService(mapPlanetMethodsToProps),

withData,

withChildFunction(renderName),

)(BodySelect)

const StarshipList =

compose(

withSwapiService(mapStarshipMethodsToProps),

withData,

withChildFunction(renderModelAndName)

)(BodySelect)

export {

PersonList,

PlanetList,

StarshipList,

}

// body-select.js

import React from "react"

import PropTypes from 'prop-types'

import Container from "../container"

const BodySelect = (props) => {

const { data, onItemSelected, children: renderLabel } = props

const itemListContent = data ? data.map((item) => {

const { id } = item

const label = renderLabel(item)

return (

<button type="button"

className="list-group-item list-group-item-action elegant-color"

key={id}

onClick={() => {

onItemSelected(id)

}}>

{label}

</button>

)

}) : null

return (

<Container content={itemListContent} />

)

}

BodySelect.defaultProps = {

onItemSelected: () => {

}

}

BodySelect.propTypes = {

onItemSelected: PropTypes.func,

data: PropTypes.arrayOf(PropTypes.object).isRequired,

children: PropTypes.func.isRequired

}

export default BodySelect

// django-swapi-service.js

export default class DjangoSwapiService {

\_apiBase = 'http://localhost:8000/api/v1'

\_imageBase = 'http://localhost:8000/media/image'

\_transformData = ({ id, ...items }) => {

return {

id: id.toString(),

...items

}

}

getResource = async (url) => {

const response = await fetch(`${this.\_apiBase}${url}`)

if (!response.ok) {

throw new Error(`Could not fetch ${this.\_apiBase}${url}, received ${response.status}`)

}

return await response.json()

}

// People

getAllPeople = async () => {

const res = await this.getResource(`/people/`)

return res.map(this.\_transformData)

}

getPerson = async (id) => {

const person = await this.getResource(`/people/${id}/`)

return this.\_transformData(person)

}

getPersonImage = ({ id }) => {

return `${this.\_imageBase}/people/${id}.jpg`

}

// Planets

getALlPlanets = async () => {

const res = await this.getResource(`/planets/`)

return res.map(this.\_transformData)

}

getPlanet = async (id) => {

const planet = await this.getResource(`/planets/${id}/`)

return this.\_transformData(planet)

}

getPlanetImage = ({ id }) => {

return `${this.\_imageBase}/planets/${id}.jpg`

}

// Starships

getALlStarships = async () => {

const res = await this.getResource(`/starships/`)

return res.map(this.\_transformData)

}

getStarship = async (id) => {

const starship = await this.getResource(`/starships/${id}/`)

return this.\_transformData(starship)

}

getStarshipImage = ({ id }) => {

return `${this.\_imageBase}/starships/${id}.jpg`

}

}