Adailson de Almeida dos Santos1; André Vinicius Vieira Mildemberg1

*1 Graduando(a) do curso de Sistemas de Informação do Centro Universitário UNIFACEAR*

**RESUMO**

*Realizamos o desenvolvimento de um software mobile de gerenciamento de horas complementares chamado DaHora, o sistema foi desenvolvido inteiramente com Android nativo utilizando a linguagem Kotlin, além disso criamos uma API em C# que é a responsável por realizar as operações com o banco de dados.*

*Utilizamos diversas metodologias para planejamento e execução do projeto a fim de tornarmos o desenvolvimento mais objetivo e conseguirmos dessa forma atender todos os requisitos elencados, entre essas metodologias podemos elencar o levantamento de requisitos funcionais e não funcionais, criação de diagramas de casos de uso, classes e entidade-relacionamento (DER).*

*Palavras-chave: Horas, Complementares, Mobile.*

***ABSTRACT***

*We carried out the development of a complementary mobile time management software called DaHora, the system was developed entirely with native android using Kotlin. In addition, we created na API in C# that is responsible for performing operations with the database*

*We use several methodologies for project planning and execution in order to make the development more objective and thus manage to meet the listed requirements, among these methodologies we can choose to collect non-functional requirements, create functional diagrams of use cases, classes and entities -relationship (DER).*

*Keywords: Hours, Complementary, Mobile.*

**1. INTRODUÇÃO**

Hodiernamente, no setor acadêmico existem duas grandes problemáticas que são a dificuldade no gerenciamento de validação e registro das horas complementares realizado pelos coordenadores de faculdades, além da visualização e acompanhamento dessas horas por parte dos alunos.

Uma vez que a gestão das horas muitas vezes fica centralizada aos coordenadores de curso os quais devem receber comprovantes, validar e registrar as horas de dezenas ou até mesmo centenas de alunos, tal quantidade pode fazer com que aconteçam erros nesse processo.

Ademais, a outra problemática elencada que é a dificuldade de visualização por parte dos alunos é deletéria, uma vez que por ter sido introduzida pela Portaria MEC nº 1.886, de 1994, e ser regulamentada atualmente pela lei 9.394/96, deve ser garantido ao aluno o direito de visualização dessas informações de maneira fácil e simplificada, pois, atualmente muitas vezes o aluno acaba tendo dificuldades em realizar a administração dessas horas por conta da falta de transparência.

Dessa forma, resolvemos criar o *DaHora* um sistema mobile de gestão de horas complementares com o intuito de ser uma solução com um melhor custo-benefício que os concorrentes além de oferecer uma maior simplicidade na sua utilização, a fim de facilitar seu uso no cotidiano. O nosso software foi criado para suprir a carência que instituições de ensino tem de ter um sistema específico e unificado para o gerenciamento de horas complementares de seus alunos onde por meio dele eles possam receber, analisar e aprovar solicitações de horas complementares, além disso os alunos também conseguirão por meio dele ver suas horas de forma clara. Dessa maneira, iremos tornar o processo de gestão de horas complementares mais eficiente para ambas as partes.

**2. DESENVOLVIMENTO**

Para este projeto nós desenvolvemos um software mobile de gestão de horas complementares desenvolvido em Android nativo utilizando a linguagem Kotlin, também desenvolvemos uma API em C# para comunicação com o banco de dados. Ademais, para nos auxiliar no desenvolvimento nos utilizamos as IDEs Android Studio, Visual Studio e o editor de código fonte Visual Studio Code.

Além disso, no processo de desenvolvimento nós utilizamos diagramas baseados na UML para definirmos de forma mais concreta o que o sistema deverá conter, para desta forma, sermos mais produtivos no desenvolvimento. Os diagramas utilizados foram o diagrama de casos de uso, diagrama de entidade relacionamento (DER), diagrama de classes entre outros.

No tópico de desenvolvimento abordaremos no Tópico 2.1 o PMCanvas do projeto. Enquanto no tópico 2.2 veremos o modelo de processo de negócio principal do aplicativo. Nos tópicos 2.3 e 2.4 veremos respectivamente os requisitos funcionais e não funcionais do programa. No 2.5 as regras de negócios da aplicação. Enquanto, nos tópicos 2.6 e 2.7 veremos o diagrama de casos de uso e o diagrama de classes. Já no tópico 2.8 teremos a descrição do caso de uso de criar um requerimento. E por fim no tópico 2.9 o diagrama de entidade-relacionamento e finalmente no tópico 2.10 falaremos um pouco sobre as tecnologias utilizadas no processo de desenvolvimento.

2.1 PMCANVAS

Para organização do projeto foi utilizado o Project Model Canvas (PMCanvas) que é uma metodologia de gerenciamento de projetos focada na agilidade e flexibilidade podendo ser aplicado no desenvolvimento de diversos tipos de projetos o PMCanvas foi desenvolvido pelo professor José Finnochio.

Segundo OSTERWALDER, o PM Canvas funciona como uma espécie de guia para empreendedores que buscam [revolucionar os seus negócios](https://www.projectbuilder.com.br/blog/o-que-e-gestao-da-mudanca-e-por-que-isso-importa-para-seu-negocio/) a partir da utilização de metodologias inovadoras e que proporcionem uma melhor articulação estratégica.

A seguir na figura 1, o PM Canvas do projeto DaHora:

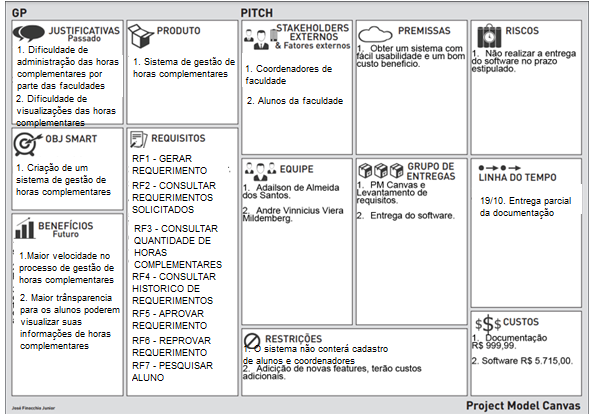


FIGURA 1: PM CANVAS  
FONTE: OS AUTORES (2022)

2.2 MODELO DE PROCESSO DE NEGÓCIO

De acordo com o autor VEYRAT Modelagem de processos de negócios é a representação gráfica do processo e todos os seus passos. Faz parte de uma metodologia chamada Business Process Management – Gerenciamento de Processos de Negócios – e é essencial para o crescimento de uma empresa. A seguir na figura 2 está sendo representado o modelo de processo de negócio da nossa aplicação.

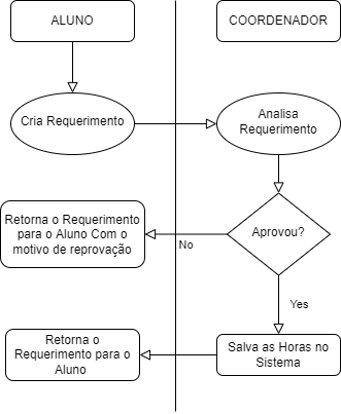


FIGURA 2: MODELO DE PROCESSO DE NEGOCIOS  
FONTE: OS AUTORES (2022)

2.3 REQUISITOS FUNCIONAIS

Segundo o autor VENTURA, quando falamos de um Requisito Funcional estamos nos referindo à requisição de uma função que um software deverá atender/realizar. Ou seja, exigência, solicitação, desejo, necessidade, que um software deverá materializar.

É fundamental para qualquer sistema o levantamento de requisitos e sua organização, o levantamento de requisitos na maioria das vezes é feito no início do processo de desenvolvimento, é ele que guia todo o resto do desenvolvimento. Além disso, é importante ressaltar que todo requisito funcional precisa ter ao menos um requisito não funcional relacionado a ele, os requisitos não funcionais serão vistos no item 2.4

|  |  |
| --- | --- |
| **Requisitos Funcionais** | **Descrição** |
| RF1 Gerar Requerimento | O aluno deve ser capaz de gerar requerimentos, onde ele deverá informar os campos: Título, data de início da atividade, data de fim da atividade, carga horária, comentários sobre a atividade, nome da instituição que a atividade será realizada. Onde após ser criado o requerimento será enviado para o coordenador do respectivo curso avaliá-lo. |
| RF2 Consultar requerimentos solicitados | O sistema deve permitir ao aluno consultar os requerimentos que ele já criou e suas respectivas situações, além das informações sobre as suas datas de criação e aprovações. |
| RF3 Consultar quantidade de horas complementares | O sistema deve possibilitar ao aluno consultar o número consolidado de horas complementares que ele já realizou. |
| RF4 Consultar histórico de requerimentos | O sistema deve possibilitar aos coordenadores de curso, consultarem o histórico de todos os requerimentos já criados pelos alunos do seu curso incluindo os que já foram aprovados e os que ainda não foram. |
| RF5 Aprovar requerimento | O sistema possibilitará ao coordenador aprovar o requerimento de horas complementares de um aluno, após realizar essa aprovação o sistema deverá salvar esse requerimento como aprovado além de adicionar as horas desse requerimento ao perfil do aluno. |
| RF6 Reprovar requerimento | O sistema possibilitará ao coordenador reprovar o requerimento de horas complementares de um aluno informando uma justificativa para essa reprovação, após realizar essa reprovação o sistema deverá salvar esse requerimento como reprovado. |
| RF7 Pesquisar aluno | O sistema possibilitará ao coordenador pesquisar pelos seus alunos e visualizar suas informações de horas complementares. |

TABELA 1: REQUISITOS FUNCIONAIS  
FONTE: OS AUTORES (2022)

2.4 REQUISITOS NÃO FUNCIONAIS

De acordo com o autor VENTURA, Um Requisito Funcional define o que o sistema fará, e o Requisito Não-Funcional define como o sistema fará.

Diante dessa afirmação, observa-se que os requisitos não-funcionais são tão importantes quanto os requisitos funcionais, pois são eles que limitam as funcionalidades e garantem a segurança do sistema. A seguir na tabela 2 está representado os requisitos não funcionais do nosso sistema

|  |  |
| --- | --- |
| **Requisito não funcional** | **Descrição** |
| RNF 1.1 Obrigatoriedade de preenchimento dos campos necessários. | Para criar um requerimento o aluno deve informar obrigatoriamente os campos: data inicial e final de realização da atividade, título, carga horária e nome da instituição que o mesmo realizou a atividade |
| RNF 1.2 Quantidade mínima de horas atingidas | Caso o aluno já tenha atingido a quantidade mínima de horas para ser aprovado no seu curso o sistema o impedirá de criar novos requerimentos |
| RNF 2.1 Restrição de acesso | O aluno poderá acessar somente os seus próprios requerimentos |
| RNF 3.1 Exibição somente de horas já aprovadas | O sistema exibirá somente as horas que já foram validadas e aprovadas. |
| RNF 4.1 Validação de acesso | O coordenador só poderá visualizar o histórico de requerimento somente dos alunos do seu curso. |
| RNF 4.2 Imutabilidade dos dados | O coordenador não poderá alterar os registros que ele já aprovou/reprovou, poderá somente visualizar para dessa forma manter a integridade dos requerimentos. |
| RNF 5.1 Registro do momento de aprovação | O sistema deverá registrar o momento que o coordenador realizou a aprovação, para consultas futuras. |
| RNF 6.1 Registro do momento de reprovação | O sistema deverá registrar o momento que o coordenador realizou a reprovação, para consultas futuras. |
| RNF 7.1 Restrição de acesso | O coordenador poderá acessar as informações de horas complementares somente dos alunos do seu curso. |
| RNF 7.2 Destaque para alunos que já concluíram suas quantidades mínimas de horas complementares. | O sistema destacará os registros de alunos que já concluíram suas quantidades mínimas de horas complementares para dessa maneira facilitar a usabilidade para o coordenador. |

TABELA 2: REQUISISTOS NÃO FUNCIONAIS  
FONTE: OS AUTORES (2022)

2.5 REGRAS DE NEGÓCIO

De acordo com OLIVEIRA CEO da BPM (Business Process Management) define regras de negócio da seguinte forma: Como o próprio termo sugere, são regras que servem para definir ou restringir alguma ação nos processos de sua empresa. São declarações que irão descrever como determinadas operações devem ser realizadas e se há algum limite que precisa ser aplicado.

Em linhas gerais, pode se considerar como regra de negócio uma regra própria da empresa que existiria mesmo se não houvesse o sistema

A seguir seguem as regras de negócio do sistema na tabela 3:

|  |
| --- |
| RN1 Após os requerimentos serem aprovados ou reprovados, suas informações não poderão ser alteradas, para assim preservar a integridade dos dados e deixa-los salvos para consultas futuras |
| RN2 Alunos que já atingiram a quantidade de horas mínima do seu respectivo curso, não poderão mais gerar requerimentos. |
| RN3 Para criar um requerimento o aluno deve informar obrigatoriamente os campos: data inicial e final de realização da atividade, título, carga horária e nome da instituição que o mesmo realizou a atividade |
| RN4 Caso o coordenador deseje reprovar um requerimento ele precisará informar um motivo para essa reprovação o qual posteriormente poderá ser acessado pelo aluno. |
| RN5 O sistema de login será feito baseado na base de dados já existente da faculdade |
| RN6 O sistema não contemplará o cadastro de novos coordenadores, cursos e alunos. |

TABELA 3: REGRA DE NEGÓCIOS  
FONTE: OS AUTORES (2022)

2.6 DIAGRAMA DE CASOS DE USO

Um diagrama de casos de uso é um diagrama dinâmico ou de comportamento na UML (Linguagem de Modelagem Unificada). O cientista de computação JACOBSON descreve casos de uso da seguinte forma, podemos dizer que um caso de uso é um documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo.

Com base nesta descrição desenvolvemos o diagrama de casos de uso do projeto tentando incluir todas os casos de uso do projeto.

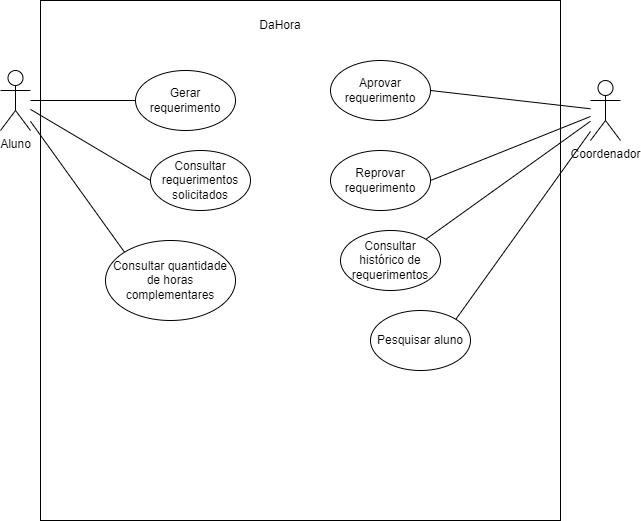


FIGURA 3: CASOS DE USO  
FONTE: OS AUTORES (2022)

2.7 DIAGRAMA DE CLASSES

Segundo BELL no livro Fundamentos básicos de UML: “O propósito do diagrama de classes é mostrar os tipos que estão sendo modelados no sistema e a relações entre eles”. Nessa perspectiva observa-se que é fulcral a elaboração de um diagrama de classes para um sistema afim de documentar corretamente o funcionamento do mesmo e facilitar futuras manutenções no código.

Na figura 4, temos representado a versão completa do diagrama de classes do projeto:

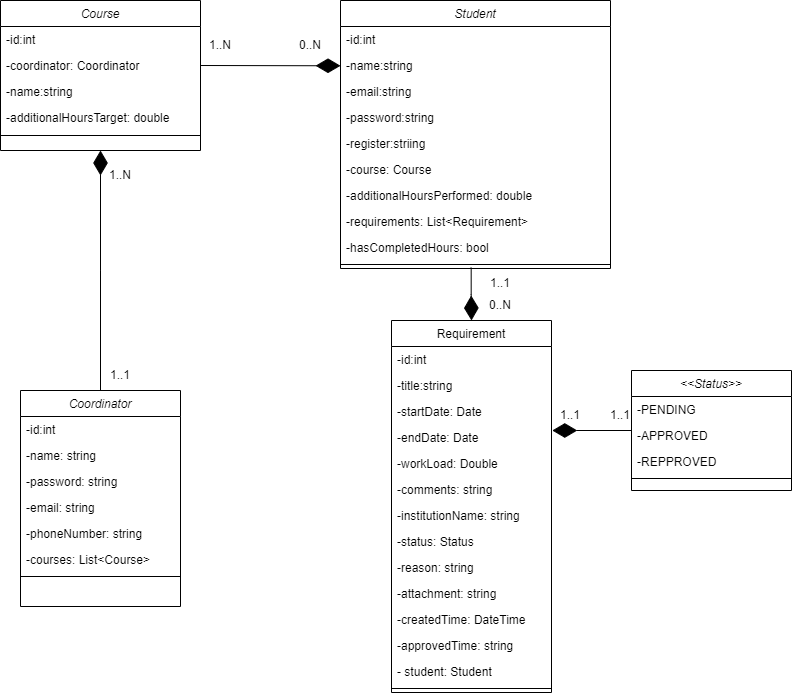


FIGURA 4: DIAGRAMA DE CLASSES  
FONTE: OS AUTORES (2022)

2.8 PROTÓTIPO E DESCRIÇÃO DE CASOS DE USO

Para o melhor entendimento sobre os casos de uso foi realizado a descrição do caso de uso de gerar requerimento, que está representado no quadro 4. Além disso, foi demonstrado o protótipo inicial da tela de gerar requerimento no item 2.9.1.

|  |  |
| --- | --- |
| Caso de Uso | CS01 – Gerar requerimento |
| Referência | Requisito funcional RF01 |
| Descrição | Permitir ao aluno efetuar a criação de Requerimento de horas complementares, com diversas informações como: Quantidade de horas, instituição, anexo de comprovante, observação, data de realização e título. |
| Atores | Aluno |
| Pré-Condições | Usuário deve estar logado no sistema e ter o perfil do tipo aluno. E deve preencher todos os campos obrigatórios. |
| Pós – Condições | Requerimento enviado para o coordenador do aluno realizar a aprovação |
| Fluxo Básico (Principal) | 1. Aluno entrar no sistema com credencias válidas 2. Seleciona a opção de gerar requerimento 3. Preenche as informações necessárias 4. Salva o Requerimento |
| Restrições/Validações | Campos obrigatórios. |
| Fluxo Alternativo | (2) Aluno já completou as horas necessárias para aprovação no curso. |
|  | O sistema não permitirá ao aluno gerar o requerimento e exibirá uma mensagem para o mesmo. |

TABELA 4: CASO DE USO GERAR REQUERIMENTO  
FONTE: OS AUTORES (2022)

2.8.1 PROTÓTIPO DA TELA MANTER COTAÇÃO

Abaixo na figura 5 está representado o Protótipo inicial da tela de criar requisição, nesta tela o usuário do tipo aluno poderá criar um requerimento que será enviado para o seu coordenador aprovar.



FIGURA 5: PROTÓTIPO DA TELA CRIAR REQUISIÇÃO  
FONTE: OS AUTORES (2022)

2.9 DIAGRAMA DE ENTIDADE RELACIONAMENTO

O MER (Modelo Entidade-Relacionamento) é uma linguagem que descreve modelos conceituais de banco de dados. Foi criada por cientista da computação Peter Chen em 1976. Por meio dela é possível representar conceitos como entidades e relacionamentos. Em contra partida, originado a partir do MER, o Diagrama Entidade Relacionamento (DER) é a sua representação gráfica e principal ferramenta utilizada nos dias de hoje para representar modelos de banco de dados.

A seguir na figura 6, está representado o Diagrama Entidade Relacionamento do software DaHora.

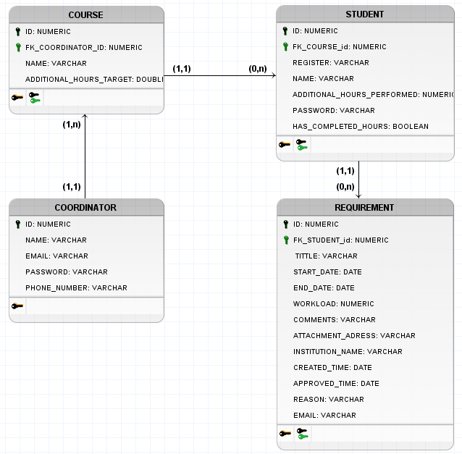


FIGURA 6: DER (DIAGRAMA ENTIDADE RELACIONAMENTO)  
FONTE: OS AUTORES (2022)

2.10 TECNOLOGIAS UTILIZADAS

Para o desenvolvimento do projeto nós fizemos uso de diversas ferramentas a seguir iremos apresenta-las e falar um pouco sobre como o uso delas nos auxiliou no desenvolvimento do projeto.

2.10.1 Android Studio

Para a programação do software nós utilizamos o Android Studio que é uma IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado) que é focada em desenvolvimento android nativo, além disso ele fornece análise de código, um depurador gráfico, um testador de unidade integrado, integração com [sistemas de controle de versão](https://en.wikipedia.org/wiki/Revision_control) (VCSes) e disponibiliza também um emulador Android para efetuar os testes de sua aplicação.

2.10.2 Draw.io

Criado em setembro de 2016, o Draw.io é um software online grátis usado para criar diagramas. Nele nós elaboramos os diagramas de classe e de casos de uso do projeto.

2.10.3 BrModelo

No processo de desenvolvimento da modelagem de dados utilizamos o software open source brModelo. Software criado como trabalho de conclusão de curso de especialização de banco de dados pelas universidades UFSC(SC) e UNIVAG (MT), orientadas pelo Prof. Ronaldo dos Santos Mello após se constatar a inexistência de uma ferramenta nacional que pudesse ser utilizada para essa finalidade.

2.10.4 Visual Studio

Para desenvolver a API C# que é responsável por fazer a comunicação com o banco de dados nós utilizamos a IDE Visual Studio da Microsoft. Ele oferece diversas ferramentas que auxiliam muito no processo de desenvolvimento.

2.10.5 Heroku

Para hospedar a API que faz conexão com o banco nós utilizamos o plataforma Heroku que oferece um plano de hospedagem gratuita, entretanto, por o Heroku não possuir um sistema de hospedagem nativo para aplicações .NET em C# nós tivemos que utilizar o Docker, o qual será abordado no tópico 2.11.5.

2.10.6 Docker

O Docker é um sistema operacional de containers onde a partir dele nós conseguimos criar maquinas virtuais chamadas de containers, que são capazes de executar aplicações de forma isolada.

Como foi mencionado no tópico 2.10.5 a plataforma Heroku não possui suporte nativo para publicação de aplicações .NET, então para contornar esse problema nós tivemos que construir um container Docker onde dentro dele é executado essa API baseado em um comando pré-definido, dessa forma conseguimos subir a aplicação no Heroku.

**3. CONCLUSÃO**

Dessa forma, através das metodologias e ferramentas apresentadas nesse artigo, Android Studio para o desenvolvimento o app mobile, Visual Studio para o desenvolvimento da API que se comunica com o banco, a elaboração dos diagramas de classes e de casos de uso para ter uma visão melhor do escopo do projeto. Além de todos os conceitos de orientação a objetos e desenvolvimento mobile vistos em aula possibilitaram o desenvolvimento e a finalização do projeto com todas as funcionalidades implementadas.

Desta forma o aplicativo DaHora consegue suprir a demanda que as faculdades tem em encontrar softwares para gestão de horas complementares de seus alunos. Melhorando assim o processo de gestão de horas, tornando o processo mais simples e seguro pois um dos pilares da construção do aplicativo foi a preservação da integridade dos registros não permitindo a alteração ou exclusão de registros no aplicativo.

**4. REFERÊNCIAS**

### BELL, DONALD. Fundamentos básicos de UML: O diagrama de classes. 1 ed. IBM Corporation. 19, dezembro de 2016

DASGUPTA**,** SANJOY**.** PAPADIMITRIOU**,** CHRISTOS**,** VAZIRANI**,** UMESH.[***Algorithms***](http://highered.mheducation.com/sites/0073523402/index.html)*.* 1 ed. Editora:  McGraw-Hill Higher Education. 13, setembro de 2006.

GUEDES, MARYLENE. **O que é MVC?** 22, junho de 2020. Disponível em: <https://www.treinaweb.com.br/blog/o-que-e-mvc/>. Acesso em: 11, de abril de 2022.

### JACOBSON, IVAN: [*Object Oriented Software Engineering: A Use Case Driven Approach*](https://www.amazon.com.br/Object-Oriented-Software-Engineering-Approach/dp/0201544350/ref=sr_1_3?qid=1585572037&refinements=p_27%3AIvar+Jacobson&s=books&sr=1-3). 1 ed. Editora: Addison Wesley. 1, julho de 1992.

### 

### 