Fala Aí! – *Feed* de notícias cooperativo

Ana Luíza Delgado de Azevedo, Dyorjenes Henrique Andrade Santos, Nikollas Ferreira Gonçalves, Tiago Alves de Oliveira, Alisson Marques da Silva

Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG)
Rua Álvares de Azevedo, 400 - Bela Vista – 35.503-822 – Divinópolis – MG – Brasil

{ana.luizadiv, dyorjenes.henrique, nikollasferreira}@hotmail.com,
{tiagofga, alissonmarques}@gmail.com

Abstract. *Communication is vital in universities, since that is a huge amount of data and usually have an extensive territory. It is seen that through a survey even with posters, murals and digital resources in some institutions are incapable to inform every student in a practical way. The proposed solution in this project is the development of an application for smartphones aimed for students and academic institutions, that its focus is CEFET-MG campus Divinópolis. This project is composed by news feeds, separated by courses offered by the institution and calendars divided by classes. The differential of this project is aimed only for students, having the functions to add, edit, rate and exclude news and events, becoming, therefore, a collaborative system.*

Resumo. *A comunicação é vital nas universidades, uma vez que estas possuem grande fluxo de informações e geralmente têm o território extenso. Nota-se através de uma pesquisa realizada que mesmo com cartazes, murais e recursos digitais como sites, a divulgação em algumas instituições de ensino é incapaz de informar todos os alunos de forma rápida e prática. A solução proposta neste projeto é o desenvolvimento de um aplicativo para dispositivos móveis voltado para os alunos de instituições acadêmicas, que tem como foco o CEFET-MG Campus Divinópolis. Sendo este composto por feeds de notícias, separados por cursos ofertados pela instituição, e por calendários divididos por turmas. O diferencial deste projeto é ser voltado unicamente para os alunos, tendo eles todas as funções de adicionar, editar, avaliar, e excluir notícias e eventos, tornando-se, portanto, um sistema colaborativo.*

1. Introdução

O ato de comunicar é inerente a qualquer ser humano. Desde a Pré-História, quando os homens passaram a viver em sociedade, eles perceberam o poder e a importância da comunicação, e então desenvolveram a linguagem fazendo-a por meio de infinitas formas (sons, gestos, cores, desenhos), sempre com o mesmo motivo: transmitir uma mensagem e ser entendido[1].

Atualmente não é possível reduzir a comunicação à transmissão de mensagens. Ela é mais do que isso, é a criação de um ambiente comum entre dois lados que participam fornecendo e extraindo informações entre eles. O qual é vital nas universidades, uma vez que estas possuem grande fluxo de informações e geralmente têm o território extenso.

Nota-se que mesmo com cartazes, murais e recursos digitais, como sites, a divulgação em algumas instituições de ensino é incapaz de informar todos os alunos de forma rápida e prática. Tal fato é apresentado na pesquisa realizada entre 95 alunos, o que representa 15,2% do total de discentes matriculados[2], no Centro Federal de Educação Tecnológica de Minas Gerais (CEFET-MG) Campus Divinópolis, que apresenta essas mesmas ferramentas de comunicação.

Nesta pesquisa, o estudante avaliou o nível de eficiência da divulgação na instituição. Analisando o resultado final nota-se que a escola não atinge todo o seu potencial de difusão das informações, pois 93,6% dos alunos consideram a qualidade da divulgação como ruim ou mediana, como pode ser ilustrado na Figura 1. Dessa forma, é necessário que algo seja feito para que a informação abranja este e os outros campi de forma eficaz.

Existem várias maneiras de resolver o problema apresentado, tais como a distribuição de panfletos, a criação de um jornal interno, murais mais chamativos, a utilização adequada e regular de sites das instituições, o uso das redes sociais como meio de comunicação, por exemplo. A solução proposta neste projeto foi o desenvolvimento, através do modelo de prototipagem de um aplicativo para dispositivos móveis voltado para o CEFET-MG Campus Divinópolis, que tenha como objetivo proporcionar um espaço de comunicação ágil entre os alunos.

Esse sistema possui um *Feed* de informações contendo tudo o que acontece na instituição, desde o cancelamento de uma prova até a realização de uma sexta-feira cultural, por exemplo, e um calendário para marcar atividades acadêmicas e afins. Sendo todas as funções gerenciadas unicamente por alunos, tornando-se, assim, um aplicativo colaborativo.

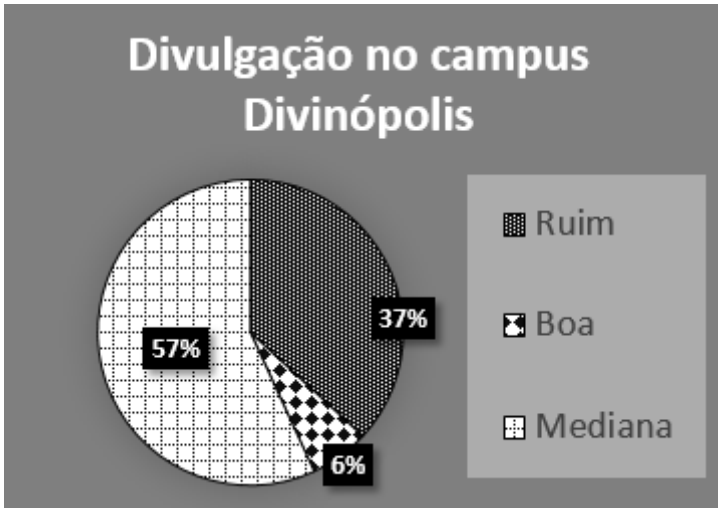


Figura 1. Avaliação da divulgação de informações dentro do CEFET-MG Campus V

2. Aplicativos Relacionados

Existem cinco aplicações que possuem funções semelhantes ao Fala Aí!, sendo três delas pagas e duas gratuitas. , dessa forma, essas semelhanças serão destacadas a seguir. Dentro das aplicações pagas, o primeiro é o ClassApp[3], que possui um calendário de eventos, o segundo é o Pertoo[4], aplicativo e software integrado de comunicação escolar, e o terceiro é o Escola Direta[5] que possui funcionalidade de mensagens e notícias. Por serem pagos não foi possível obter mais informações sobre eles.

Já os aplicativos gratuitos são o App do Instituto Nossa Senhora da Piedade (INSP) [6] que possibilita à seus usuários compartilhem mensagens (WhatsINSP) e fotos (INSPgram), além de acessarem o acervo de vídeos (INSPtube), boletins e notas (Secretaria online) e notícias, entretanto o aplicativo é voltado apenas para essa instituição. E o Remind[7] que tem por objetivo tornar os estudantes e os pais mais envolvidos nos estudos e economizar tempo dos professores, que deixam de ter de escrever recados na agenda de cada aluno ou mandar e-mails para todos os responsáveis. Contudo o aplicativo é basicamente restrito às mensagens.

Como pode ser visto, existem muitos aplicativos que se assemelham ao Fala Aí!, como os citados anteriormente, porém a grande maioria é paga e não são focados totalmente aos alunos e na agilidade em que as informações são repassadas. O diferencial do presente projeto é que ele possui um mural de notícias alimentado unicamente por alunos. Lá eles postam, visualizam e controlam notícias e eventos, já os *apps* citados são voltados para a integração de pais/escola.

3. Materiais e Métodos

Nesta seção será explicado os materiais e métodos utilizados no projeto, tais como o sistema operacional, IDE, linguagem de programação, sistema de gerenciamento de banco de dados (SGBD), WebService, e notação de objetos utilizada no projeto

3.1. Sistema Operacional

Na criação de um aplicativo é necessário escolher para qual sistema operacional (SO) ele será voltado. Com a intenção de abranger o máximo de usuários possíveis é necessário que se escolha o SO mais utilizado nos *smartphones* entre o público alvo do sistema. Para isso realizamos uma pesquisa entre 95 alunos do CEFET-MG Campus V, o que representa 15,2% do total de discentes matriculados na instituição, para avaliar qual SO mais utilizado por eles. Como pode ser visto na figura 2 o resultado foi que 83,2% dos entrevistados possuem *Android* como sistema operacional, 9,5% IOS, 6,3% Windows Phone e 1,1% utilizam outros ou não possuem *Smartphones*.

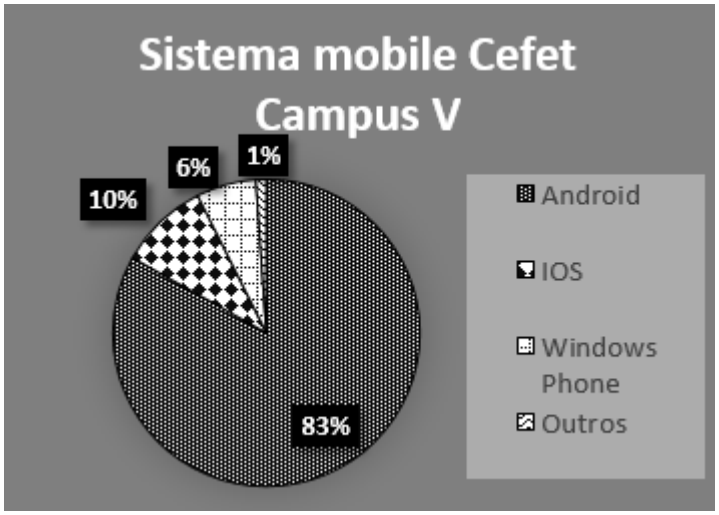


Figura 2. Sistema operacionais para celulares utilizados por alunos do CEFET-MG Campus V

3.2. Ambiente de Desenvolvimento Integrado - IDE

IDE, do inglês Integrated Development Environment ou Ambiente de Desenvolvimento Integrado, é um programa de computador que reúne características e ferramentas de apoio ao desenvolvimento de software com o objetivo de agilizar este processo. O Android Studio é uma IDE voltada para o desenvolvimento de aplicativos para o sistema operacional Android. Ele utiliza de várias linguagens como Java e C++ e de bibliotecas particulares para o desenvolvimento mobile[8]. Além de ser recomendada pela Google, uma das criadoras do sistema, uma de suas vantagens é ser específica para o sistema *Android*, tornando a ferramenta mais eficiente. E também possui uma vasta documentação e de fácil entendimento.

3.3. Linguagem de Programação

A linguagem de programação Java segue o paradigma de orientação a objetos, que é um conceito que está relacionado com a ideia de classificar, organizar e abstrair coisas do mundo real para codificação, o que a torna uma ferramenta extremamente poderosa[9].

Essa linguagem foi utilizada como base para o desenvolvimento de todo o aplicativo. Sua escolha se deve pelo suporte da IDE utilizada e por sua ampla documentação e comunidade.

3.4. Sistema de Gerenciamento de Banco de Dados

O MySQL é um banco de dados completo, robusto e extremamente rápido. Ele é multi-plataforma e uma parte de seus recursos é disponível ao público gratuitamente. A vantagem desse SGBD (Sistema de gerenciamento de banco de dados) se dá por ele ser simples, tanto para um programador iniciante quanto para um desenvolvedor mais experiente devido a sua facilidade de programação e aprendizado. Além disso, pode ser utilizado nas aplicações mais simples até as mais robustas juntamente com sua possibilidade de permitir que sejam implementadas regras de segurança no servidor.

3.5. *WebService*

Webservice é uma solução utilizada na comunicação, principalmente, entre duas plataformas diferentes por intermédio de linguagens como o JSON, que será explicado no item 3.6 No caso do nosso aplicativo, ele foi utilizado pelo fato do *Android* não ter suporte nativo do MySQL, e, visto que um aplicativo mobile normalmente não deve acessar diretamente um banco de dados para recuperar ou adicionar informações. Além disso, a aplicação não atenderia aos requisitos de comunicação se funcionasse *off-line*, portanto é necessário que ela consuma esse serviço para acessar o banco.

3.6. JSON

O JSON é uma forma de transferência e/ou intercâmbio de dados que o Java dá suporte. Uma de suas vantagens é a validação de dados, na qual todos eles são validados automaticamente, e, no caso do JSON ser inválido nada será inserido no registro. Além disso, seu acesso é eficiente e otimizado, ou seja, os documentos JSON salvos nas colunas do tipo JSON são na verdade convertidos num formato interno que permite leitura rápida aos elementos do documento[12]. Quando o servidor precisar ler um valor JSON armazenado em formato binário, o valor não precisa ser convertido a partir de uma representação em texto.

4. Desenvolvimento

O aplicativo Fala Aí! foi desenvolvido contendo funções de cadastro e *login, feeds* de notícias e um calendário. Como pode ser visto no diagrama da figura 3 seus casos de uso são: ler, avaliar e reportar notícias, e gerenciar notícias e calendário. Dentro das funções de gerenciar, tanto nas notícias quanto no calendário é possível adicionar, editar e excluir informações.

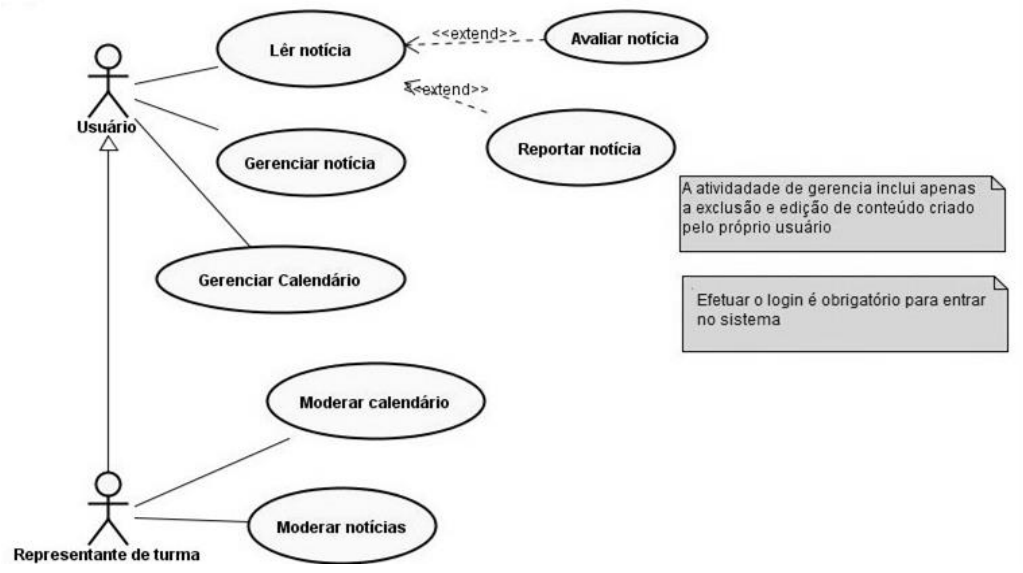


Figura 3. Diagrama de Caso de Uso

Já o processo de avalição das notícias pode ser realizado por qualquer aluno cadastrado, ele poderá avaliá-las como positivas (Fala mais) ou negativas (Fala menos). Outra questão a ser ressaltada é que os usuários são divididos em duas categorias: a de usuário comum e a de representante de turma. Todos os líderes de turma possuem as unções do usuário, com a diferença que somente eles podem moderar os *feeds* e o calendário.

Como pode ser verificado no diagrama de caso de uso, muitas são as funções do aplicativo. E para isso, foram necessárias diversas ferramentas que comuniquem entre si. Então, como o Android Studio é em Java e por já ser compatível com o JSON, é possível fazer o intermédio, por meio deste, com o *webservice*. A parte de desenvolvimento das classes em Java, feita no Android Studio, e pode ser vista no diagrama de classes, representado na Figura 4. Já os métodos principais de cada classe serão apresentados no apêndice.

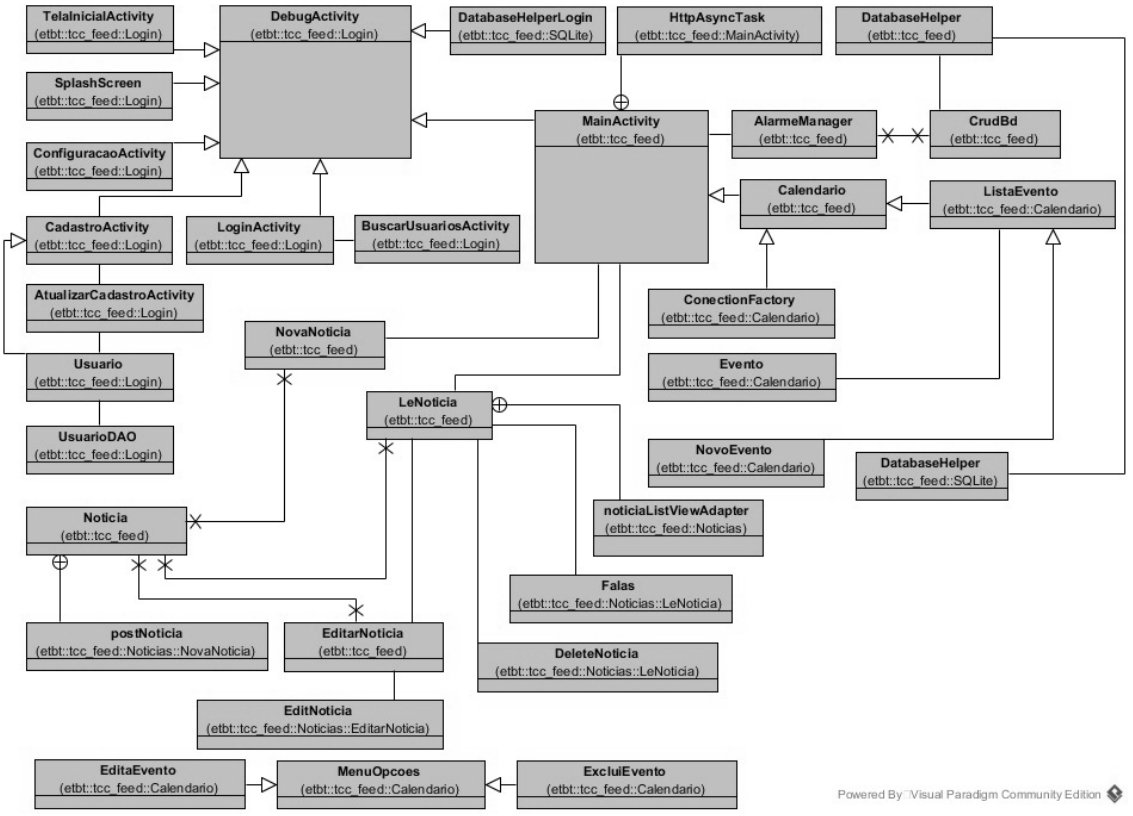


Figura 4. Diagrama de Classes

O diagrama de classes do Fala Aí! é bem extenso devido às suas funcionalidades. Como cada classe representa funções diferentes, elas serão explicadas brevemente a seguir. A classe Noticia guarda os atributos que dizem respeito a notícia que será apresentada na tela para o usuário. A MainActivity diz respeito a funcionalidade principal do aplicativo, que é a leitura de notícias em forma de lista e a interação com o menu de feeds.

Já as classes que gerenciam a notícia, como sua leitura, edição e exclusão são a NovaNoticia, EditarNoticia e LeNoticia. Os eventos do calendário têm uma representação similar com o do usuário, porém com os atributos definidos pela classe Evento e UsuarioDAO, respectivamente. Todas essas classes de evento e notícia possuem classes privadas para executar tarefas assíncronas, ou seja, em paralelo com as demais para que o sistema não tenha que esperar algo acontecer para continuar seu percurso natural.

Já a parte de desenvolvimento do banco de dados poderá ser vista pelo Modelo de Entidade e Relacionamento (MER) a seguir (Figura 5). O MER é um modelo conceitual do banco de dados para descrever os objetos, chamados de entidades, com suas características, os atributos de cada entidade, e como elas se relacionam entre si. E no caso do Fala Aí! existem três entidades representadas, Aluno, Evento e Notícia. Cada uma possui seus atributos que as identificam de forma única. E visto que devem haver relacionamentos, tanto Notícia quanto Evento possuem o idAluno para representar qual aluno fez determinada ação no sistema. O aplicativo, propriamente dito, não acessará o banco de dados diretamente, então será feita a requisição por meio do *Webservice* para disponibilizar seus diversos recursos.

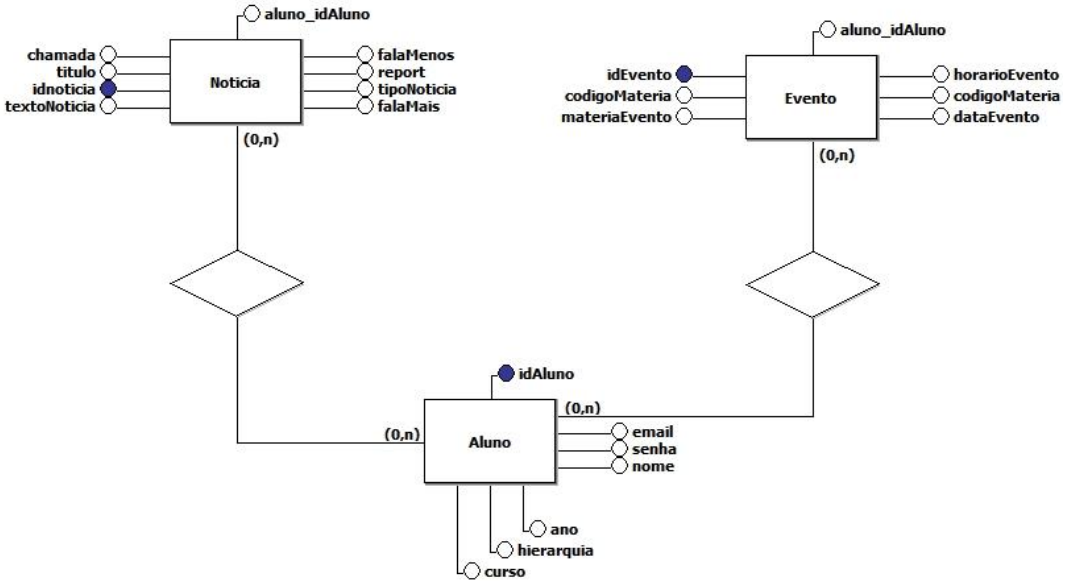


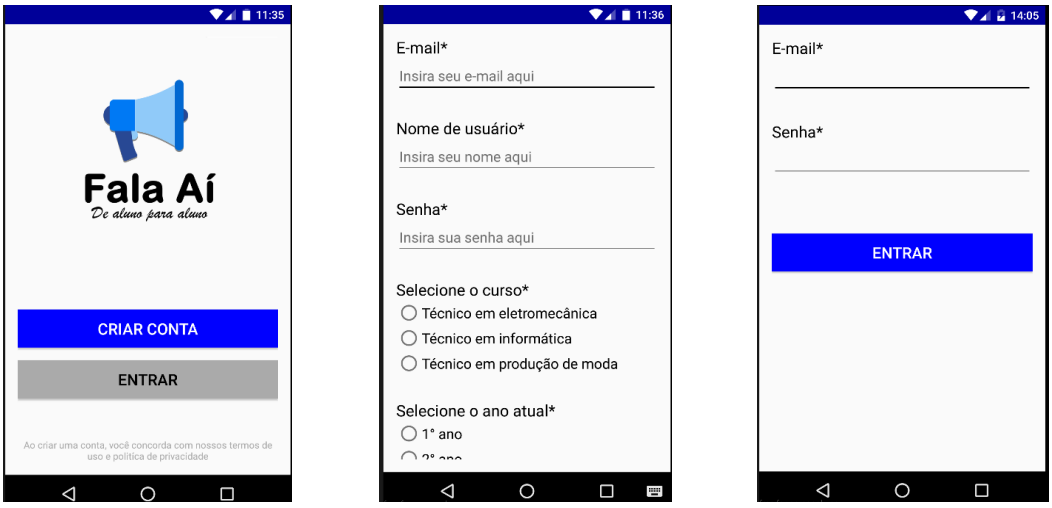
Figura 5. Modelo de Entidade e Relacionamento

Uma das principais utilidades do programa é seu sistema de notícias, e para que funcionasse no Android foram necessárias realizar requisições, por intermédio do JSON, ao *Webservice*, sendo elas: receber, adicionar, atualizar e deletar notícias. Para recebe-las, o aplicativo solicita os dados necessários, que são a id, o título, a chamada, o texto, os reports, os Fala Mais e Fala Menos (sistema de avaliação do conteúdo) de cada notícia. Com isso é convertido para um formato mais agradável ao usuário. Já para adicionar uma nova notícia é o método inverso, são recebidos dados em formato de texto e convertidos para o JSON para, assim então, fazer a requisição no *Webservice* e, caso seja válida, inseri-la no banco de dados.

Para atualizar uma notícia já existente, o processo varia tanto do recebimento dos dados quanto da volta deles ao *Webservice*, ou seja, recebe-se os dados desta e assim que o usuário finaliza volta ao processo de uma nova notícia, porém, desta vez com a inclusão de sua id. E para deletar uma notícia, é necessário que tenham pelo menos cinco *reports* e que apenas o usuário representante de turma pode excluí-la.

No caso do calendário e seus eventos, o padrão de requisições se repete, com o diferencial de que os dados necessários para serem feitas variam de acordo com os dados referentes aos eventos. E por fim, as requisições de login, que são a primeira etapa da aplicação são necessários intermédios do banco de dados SQLite para não ser preciso necessariamente de fazer todo esse processo de requisição constantemente, visto que o telefone já possui métodos para proteger seu usuário.

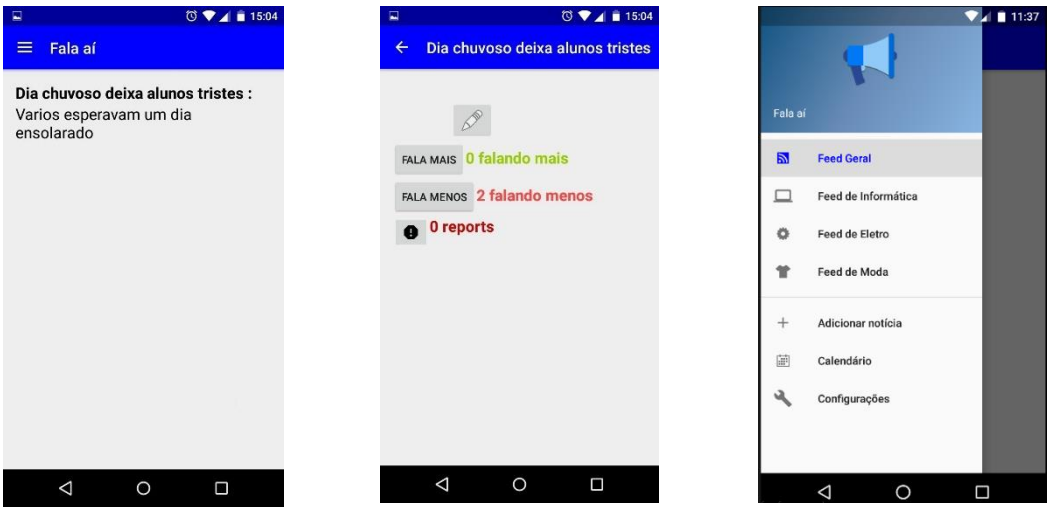
Ao termino do desenvolvimento foi construído um aplicativo em que contém uma tela inicial (Figura 6.a.) que possui a opção de cadastrar ou entrar no sistema. Caso o usuário escolha cadastrar, ele será redirecionado para uma tela de cadastro (Figura 6.b.), onde adicionará dados pessoais, como nome, curso, turma (1°, 2° ou 3° ano), e-mail, senha e se é representante ou não. O processo de identificação de cada usuário será feito através da verificação do e-mail. Não pode existir mais de um usuário com o mesmo e-mail, e para verificar se o e-mail existe foi usado uma função de validação própia do Android Studio. Caso já seja cadastrado, o usuário irá para a tela de *login* (Figura 6.c.), que possui os campos de nome e senha.



(a) **(b)** **(c)**

Figura 6 – Telas do Sistema (a) Tela Inicial, (b) Tela de Cadastro e (c) Tela de Login

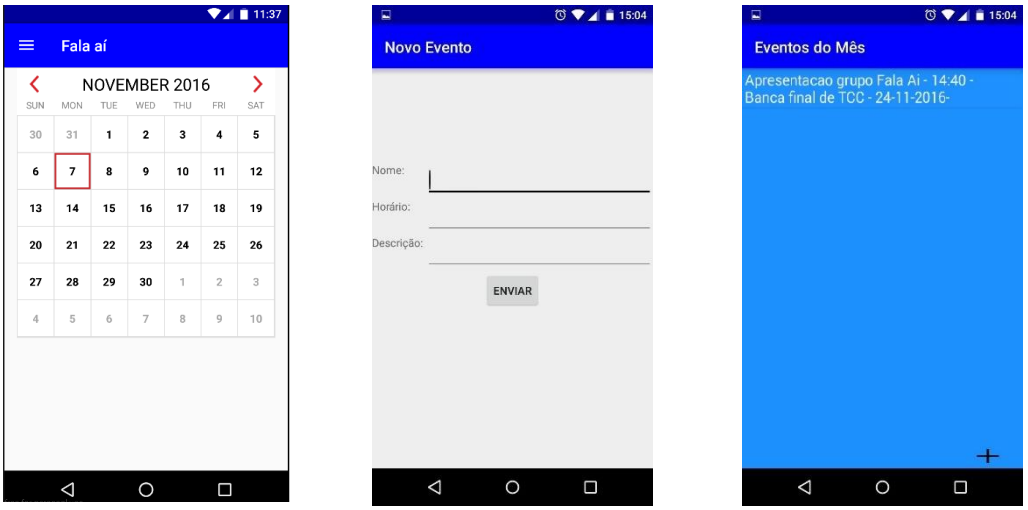
Após o login, o aplicativo é redirecionado automaticamente para a tela principal (Figura 7.a.), que é um *feed* de notícias de assuntos gerais. Ao clicar em uma notícia específica será aberta uma tela possuindo três botões para avaliação (reporte, FalaMais, FalaMenos) e maiores detalhes sobre ela (Figura 7.b.). Caso o usuário queira ver notícias de um curso específico, por exemplo produção de moda, informática ou eletromecânica, é possível acessar um feed exclusivo para cada um deles através de um menu que fica na lateral da tela (Figura 7.c.).



(a) (b) (c)

Figura 7 – Telas de Notícia (a) Feed Geral, (b) Detalhes da Notícia e (c) Barra de Menu

Este menu também possui as opções de adicionar uma nova notícia e acessar o calendário. Ao acessar o calendário (Figura 8.a.) o usuário terá acesso ao calendário do mês da data em que ele está acessando, e ao clicar em algum dia, abrirá uma nova tela (Figura 8.b.), com uma lista de eventos do mês, um botão de adicionar novos eventos, que ao clicar abrirá uma tela de adicionar evento (Figura 8.c.), e ao clicar em um evento específico, abrirá um menu com as opções de editar e excluir o evento.



(a) (b) (c)

Figura 8 – Telas do Calendário: (a) Calendário, (b) Adicionar Evento e (c) Lista de Eventos

5. Considerações Finais

Com integração da sociedade e o melhoramento da tecnologia é necessário uma constante atualização em nossos meios de comunicação. O aplicativo "Fala Aí!", com as funções de gerenciar notícias e calendário já em funcionamento, proporciona a maior rapidez e facilidade na troca de informações entre alunos, dessa forma, atingiu seu objetivo final.

Contudo, alguns empecilhos foram encontrados durante o desenvolvimento, a conciliação dos *WebServices* com o aplicativo foi um deles. Além disso, o desenvolvimento do sistema de identificação de cada aluno, tomou bastante tempo do trabalho, porém todos problemas citados foram superados e o aplicativo está funcional.

5.1 Trabalhos Futuros

Apesar de possuir todas as funções propostas no início do ano, seria interessante, em futuras alterações, a integração de professores e técnicos administrativos dentro do aplicativo. Desta forma a escola inteira estaria dentro de uma rede, tornando a comunicação entre todos mais rápida e eficaz.

Outra funcionalidade interessante para se adicionar, seria a de notificações dos eventos marcados no calendário. Sendo elas classificadas e notificadas de acordo com a sua relevância. Ou seja, cada usuário escolheria a relevância que o evento tem para ele, e receberia uma notificação de acordo com ela, por exemplo, um evento marcado como dificuldade alta envia notificações desde uma semana antes de acontecer. Ajudando, assim, todos os usuários a manterem suas agendas organizadas.

Referências Bibliográficas

[1] MANESCO, M. **A importância da comunicação para as empresas**. Disponível em: <http://www.racecomunicacao.com.br/blog/a-importancia-da-comunicacao-para-as-empresas/>. Acesso em: 07 mai. 2016.

[2] CEFET-MG campus Divinópolis. Registro Escolar.2016.

[3] CLASSAPP. **ClassApp - O aplicativo de comunicação escolar**. Disponível em: <https://www.classapp.com.br/>. Acesso em: 24 abr. 2016.

[4] PERTOO. **Pertoo - Solução Completa de Comunicação Escolar**. Disponível em: <https://www.pertoo.com/>. Acesso em: 23 abr. 2016.

[5] ESCOLA DIRETA. **Escola Direta**. Disponível em: <http://www.escoladireta.com.br/>. Acesso em: 12 mar. 2016.

[6] INSP. **INSP lança aplicativo próprio para telefones e tablets**. Disponível em: <http://www.insp2.com.br/insp-lanca-aplicativo-proprio-para-telefones/>. Acesso em: 24 abr. 2016.

[7] REMIND. **Remind**. Disponível em: <https://www.remind.com/pt-BR/>. Acesso em: 23 abr. 2016.

[8] LECHETA, R. R.; **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** 5ª Edição. Brasil: Novatec, 2015. 1072 p.

[9] **Java e Orientação a Objetos**, licença Creative Commons, 7ª edição, pela Caelum - Ensino e Inovação.

[10] MILANI, André. **MySQL:** Guia do Programador. Novatec, 2007. 400 p.

[11] LECHETA, R. R; **Web Services RESTful: Aprenda a criar web services RESTful em Java na nuvem do Google**. Brasil: Novatec, 2015. 432 p.

[12] ECMA. **The JSON Data Interchange Standard.** *Disponível* em: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf> Acesso em 20 de out. 2016.

[13] IBM. **Diagrama de objetos**. Disponível em: <http://www.ibm.com/support/knowledgecenter/pt-br/ss5jsh_9.1.2/com.ibm.xtools.modeler.doc/topics/cobjdiags.html>. Acesso em: 14 set. 2016.

[14] MICROSOFT. **Diagramas de sequência uml: referência**. Disponível em: <https://msdn.microsoft.com/pt-br/library/dd409377.aspx>. Acesso em: 14 set. 2016.

[15] FUNPAR. **Diretrizes: diagrama de colaboração**. Disponível em: <http://www.funpar.ufpr.br:8080/rup/process/modguide/md_coldm.htm>. Acesso em: 14 set. 2016.

[16] DSC UFCG. **Diagramas de estado**. Disponível em: <http://www.dsc.ufcg.edu.br/~jacques/cursos/map/html/uml/diagramas/estado/diag_estados.htm>. Acesso em: 14 set. 2016.

[17] IBM. **Diagramas de estrutura composta**. Disponível em: <http://www.ibm.com/support/knowledgecenter/pt-br/ss5jsh_9.1.1/com.ibm.xtools.modeler.doc/topics/ccompstruc.html>. Acesso em: 14 set. 2016.

[18] INTELLECTUALE. **Ambientes Integrados de Desenv**olvimento. Disponível em: <http://linguagemc.com.br/ambientes-integrados-de-desenvolvimento/>. Acesso em: 06 out. 2016.

Apêndice

Para maior detalhamento do desenvolvimento do software serão descritos a seguir alguns diagramas que representam o Fala Aí!.

1. Diagrama de Objetos

Na UML, os diagramas de objetos fornecem uma captura instantânea das instâncias em um sistema e os relacionamentos entre as instâncias. Eles usam notação semelhante à usada nos diagramas de classe. No entanto, enquanto os de classe mostram os classificadores reais e seus relacionamentos em um sistema, os diagramas de objetos mostram instâncias específicas desses classificadores e os links entre elas em um determinado momento[13]. Dentro do Fala Aí! Os objetos a serem instanciados são: Aluno, que recebe todos os dados do usuário, evento e notícia, que recebem as informações, e a avaliação de notícia, que é dividida em fala mais e fala menos. O diagrama de objeto do aplicativo (Figura 12) pode ser visto abaixo:

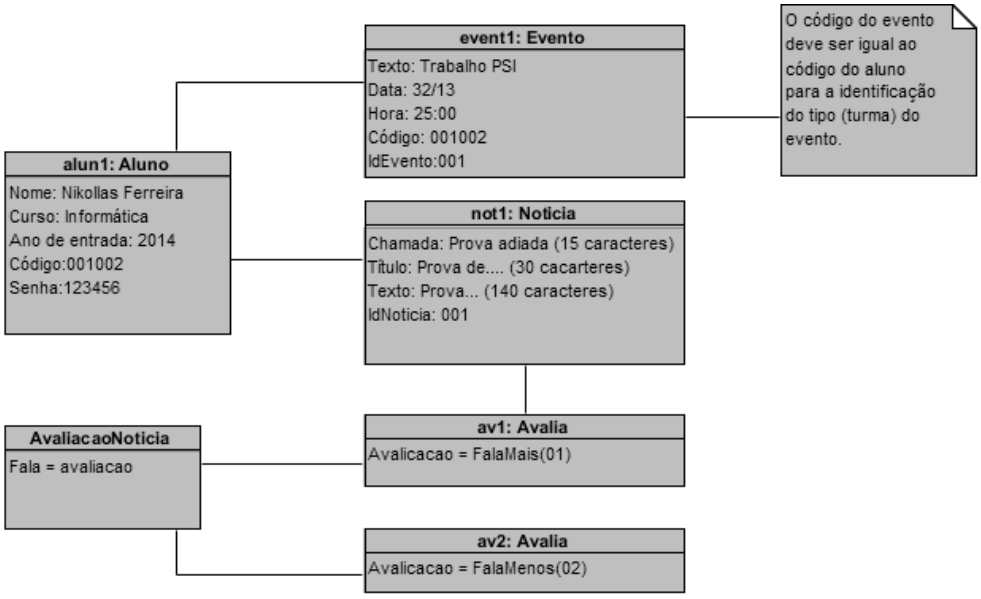


Figura 12. Diagrama de Objeto

2. Diagrama de Sequência

Um diagrama de sequência mostra uma interação, que representa a sequência de mensagens entre instâncias de classes, componentes, subsistemas ou atores. O tempo flui para baixo no diagrama e mostra o fluxo de controle de um participante para outro[14]. O diagrama de sequência do Fala Aí! (Figura 13) possui um usuário que inicia o aplicativo e percorre todo ele com a sequência de ler, adicionar, editar e depois excluir notícias. Após essas ações ele entra no calendário, vê, adiciona, edita e exclui eventos, e depois sai do aplicativo. A sequência de classes que o sistema percorre pode ser vista na Figura 13.

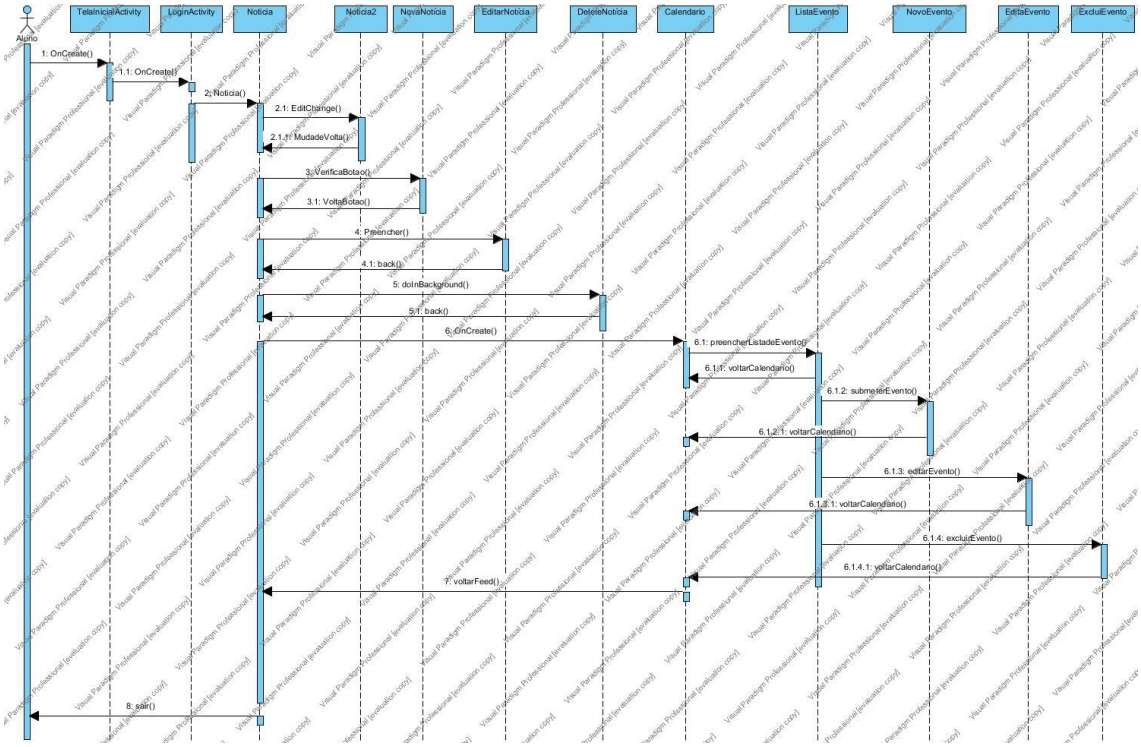


Figura 13. Diagrama de Sequência

3. Diagrama de Colaboração

Os diagramas de colaboração são usados para mostrar como os objetos interagem para executar o comportamento de um caso de uso ou de parte de um caso de uso. Ao contrário de um diagrama de sequência, o diagrama de colaboração mostra os relacionamentos entre os objetos. Ambos expressam informações semelhantes; o que muda é a forma como elas são mostradas[15]. Com a mesma sequência usada pelo usuário no diagrama de sequência, mostrado anteriormente, a interação entre os objetos no diagrama de colaboração do Fala Aí! (Figura 14) pode ser vista a seguir:

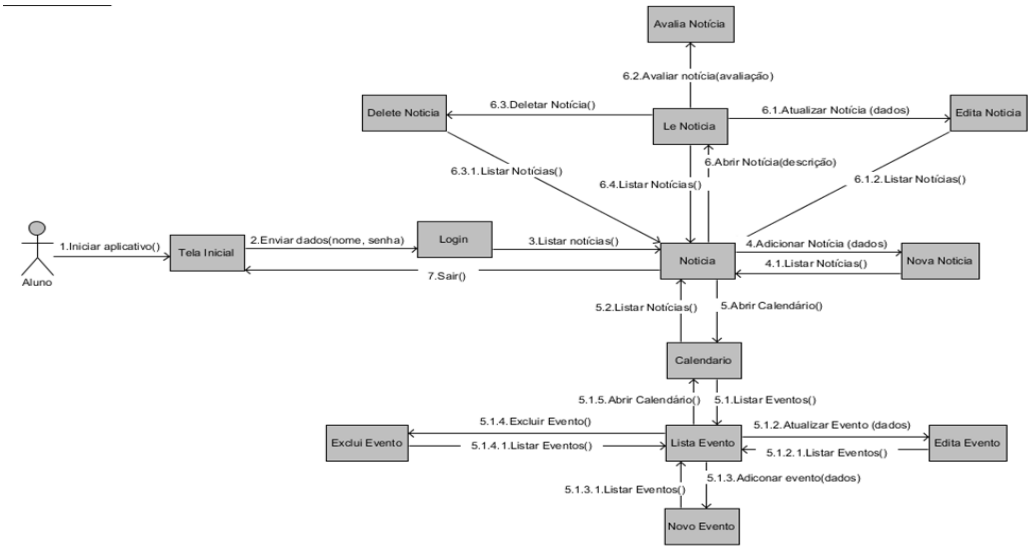


Figura 14. Diagrama de Colaboração

4. Diagrama de Estado

Em um diagrama de estado, um objeto possui um comportamento e um estado. Esse diagrama mostra os possíveis estados de um objeto e as transações responsáveis pelas suas mudanças[16]. O diagrama (Figura 15) do Fala Aí! Possui 5 estados, sendo eles: listando, gerenciando, avaliando/reportando, excluindo/editando notícia, e gerenciando calendário. Ele pode ser visto abaixo:

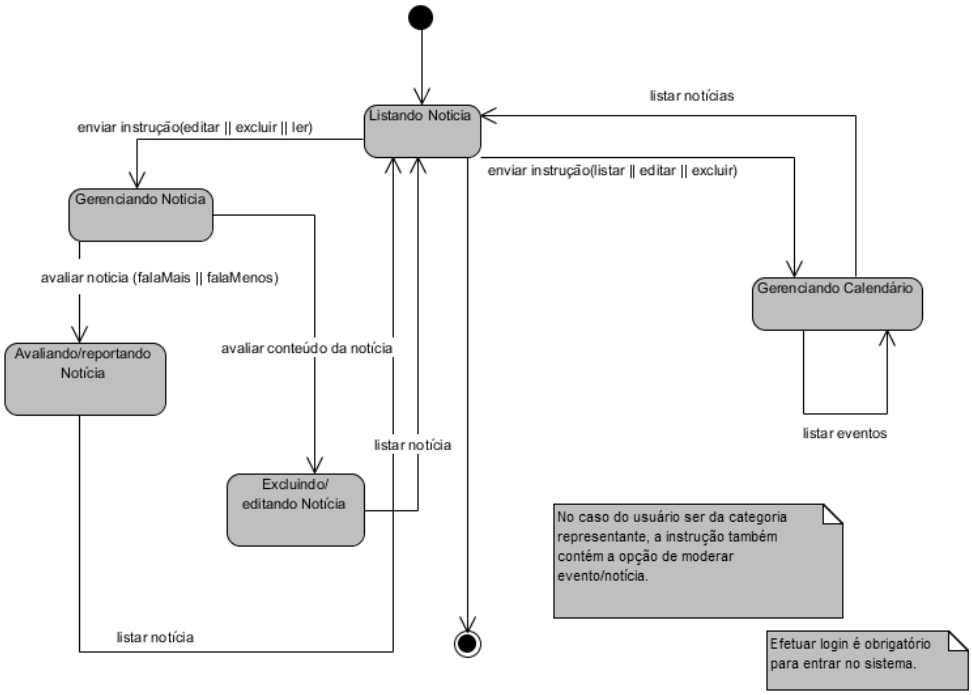


Figura 15. Diagrama de Estado

5. Diagrama de Estrutura Composta

Nos modelos UML, um diagrama de estrutura composta mostra a estrutura interna dos classificadores estruturados utilizando peças, portas e conectores. Um classificador estruturado define a implementação de um classificador e pode incluir uma classe, um componente ou um nó de implementação[17]. O diagrama de estrutura composta do aplicativo (Figura 16) foi desenvolvido através de seu caso de uso, tendo ele as funções de gerenciar calendário e notícias, e fazer login, dentro de cada uma dessas funções foi colocado as classes que elas utilizam. O diagrama pode ser visto abaixo:

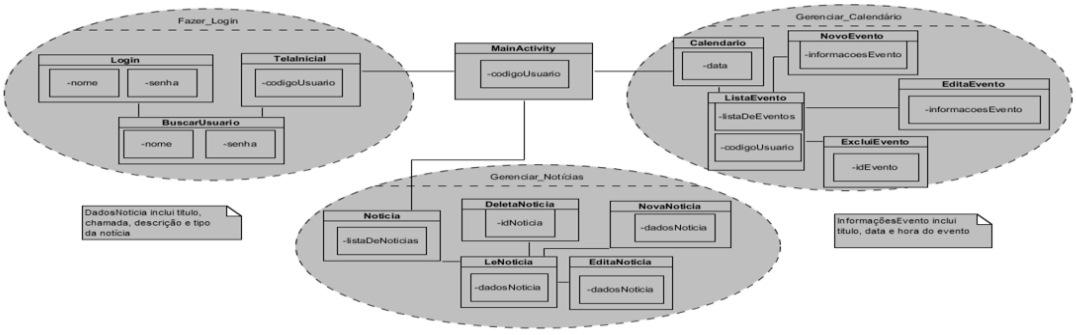


Figura 16. Diagrama de Estrutura Composta

6. Diagrama de Classes

A seguir será apresentado o diagrama de classe fragmentado do Fala Aí com seus métodos principais, dividas por funções do aplicativo. Sendo elas calendário, notícias, notificações e configurações de usuário. Salientasse que os números da figura abaixo não representam a cardinalidade do diagrama, são para fins de marcação das junções das figuras abaixo.

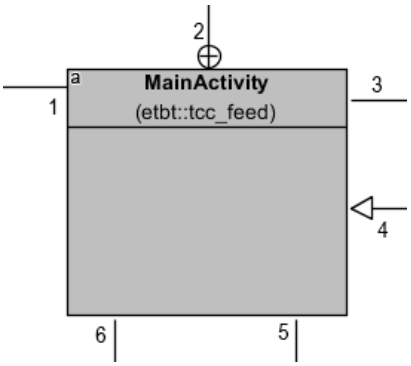


Figura 17. Classe principal do projeto. As figuras a seguir estão ligadas com essa classe, as ligações são representadas pelos números 1, 2, 3, 4, 5 e 6.

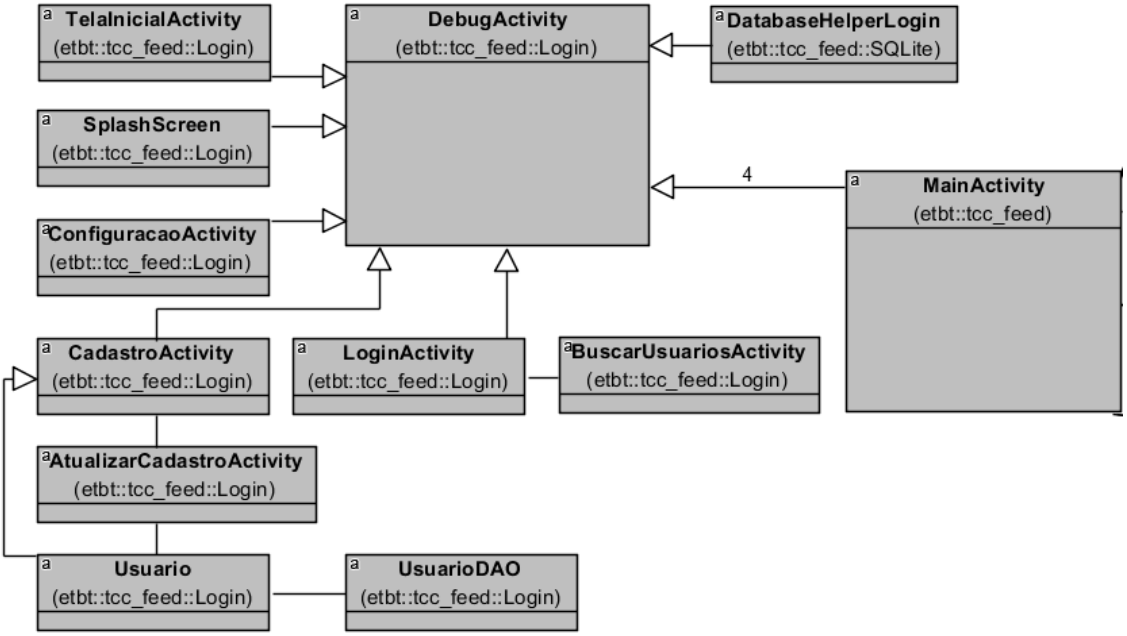


Figura 18. A Classe MainActivity relacionasse com as outras classes para realizar o cadastro, login e outras atividades do usuário.

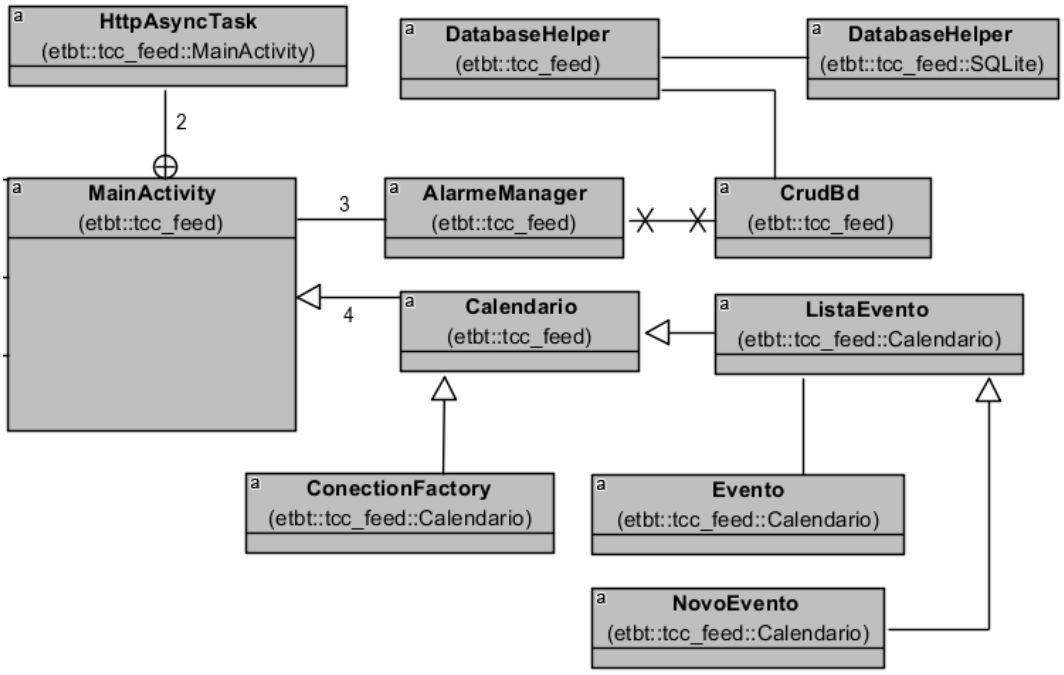


Figura 19. A Classe MainActivity relacionasse com as outras classes para realizar as funções do calendário.

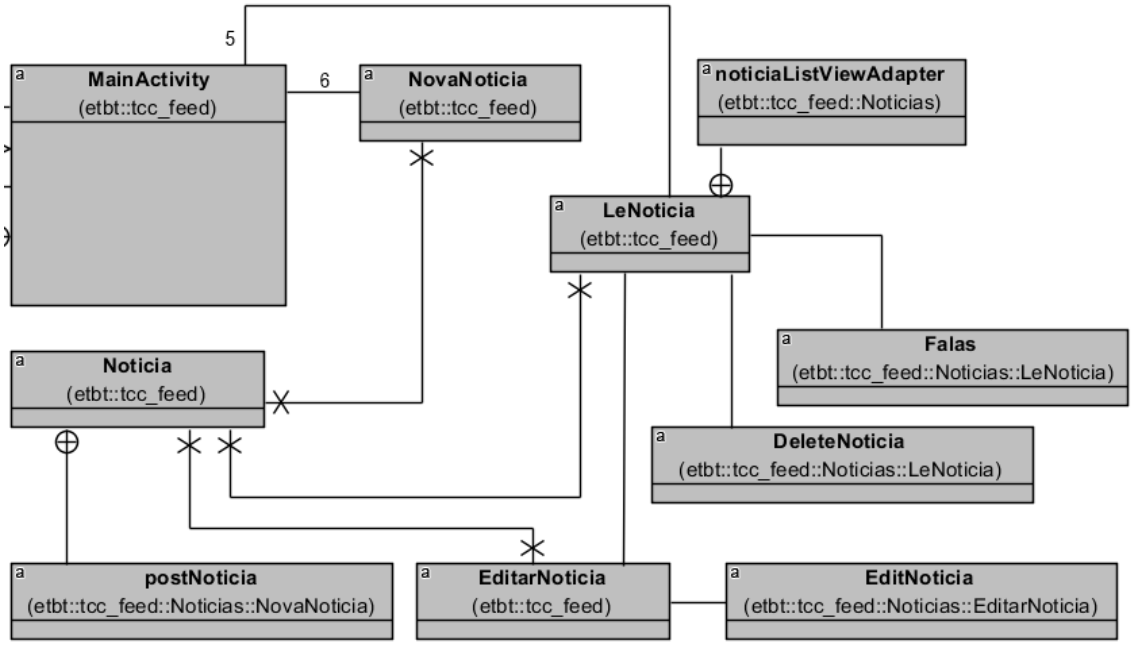


Figura 20. A Classe MainActivity relacionasse com as outras classes para realizar as funções da notícia.

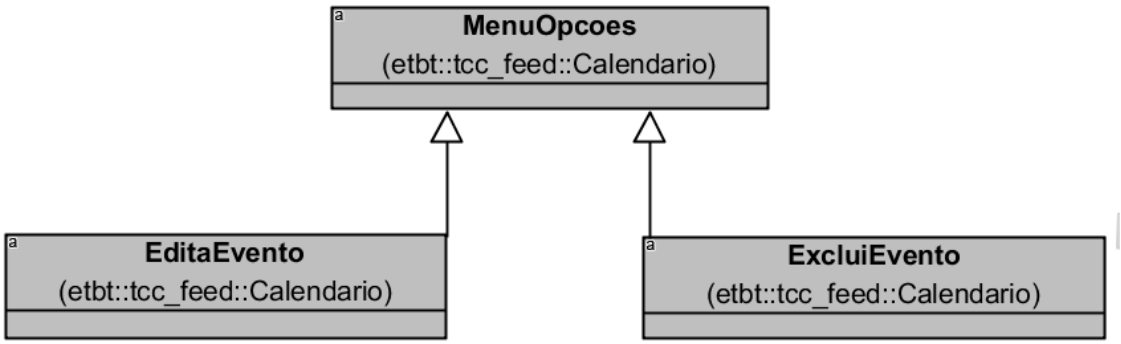
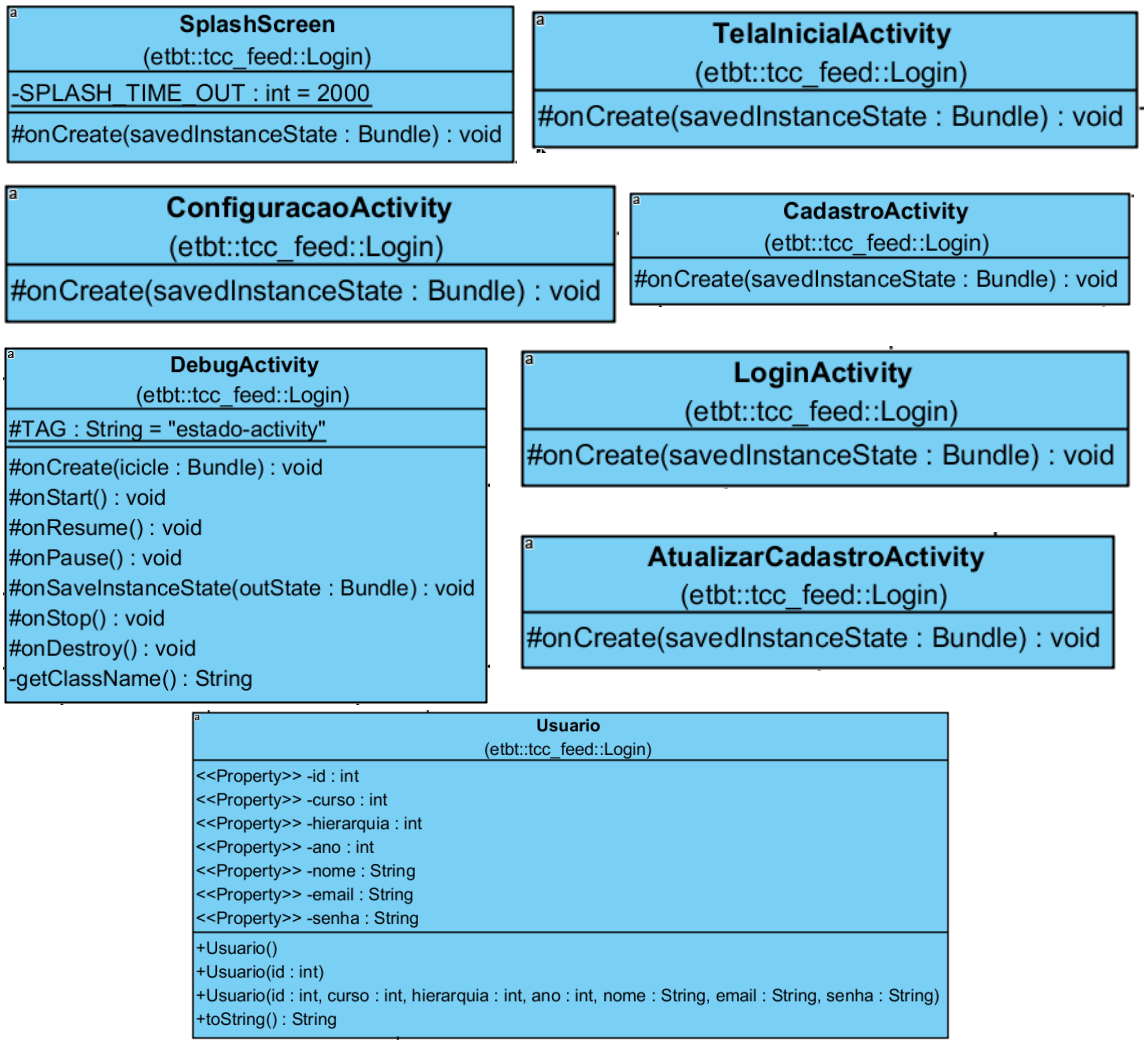
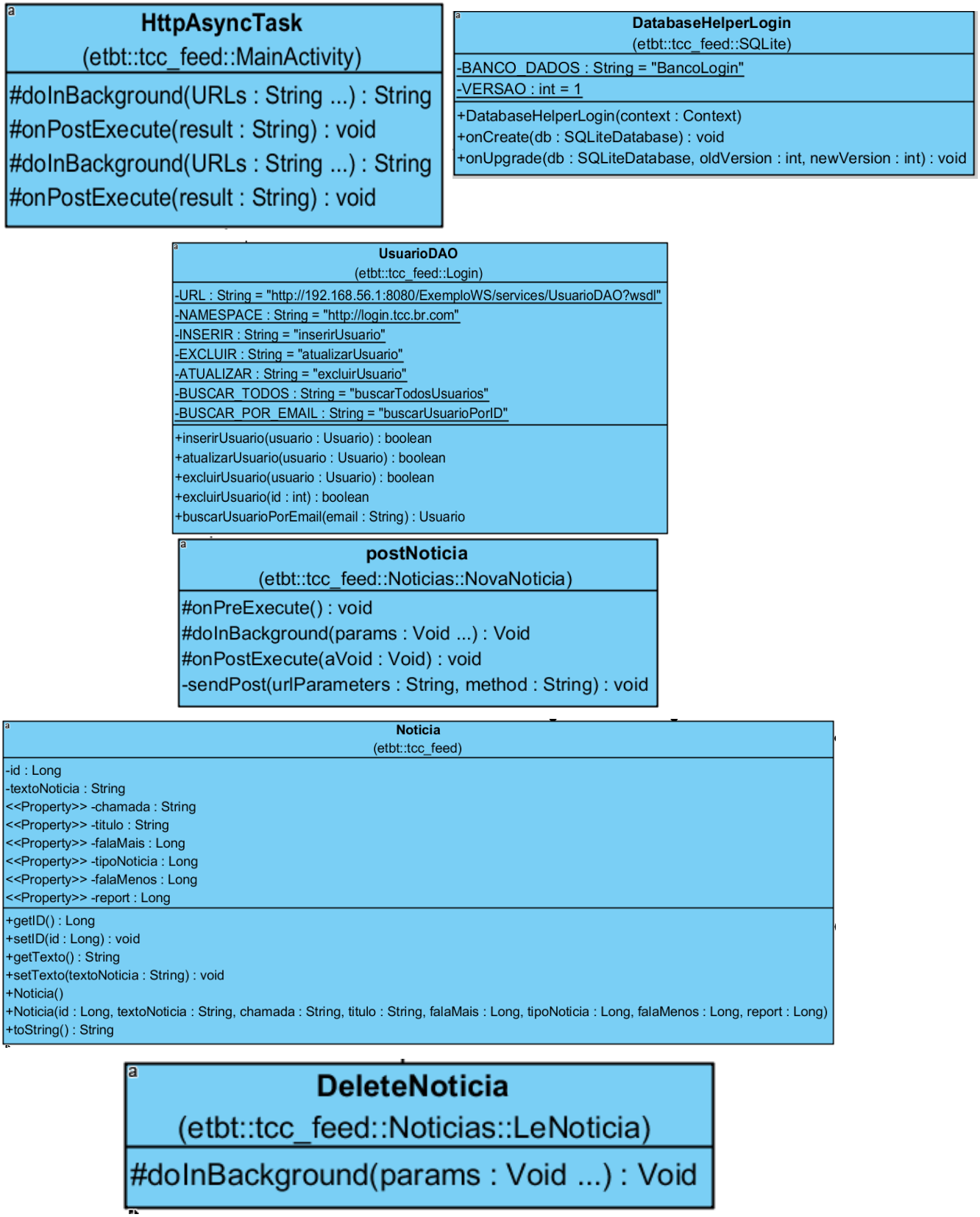


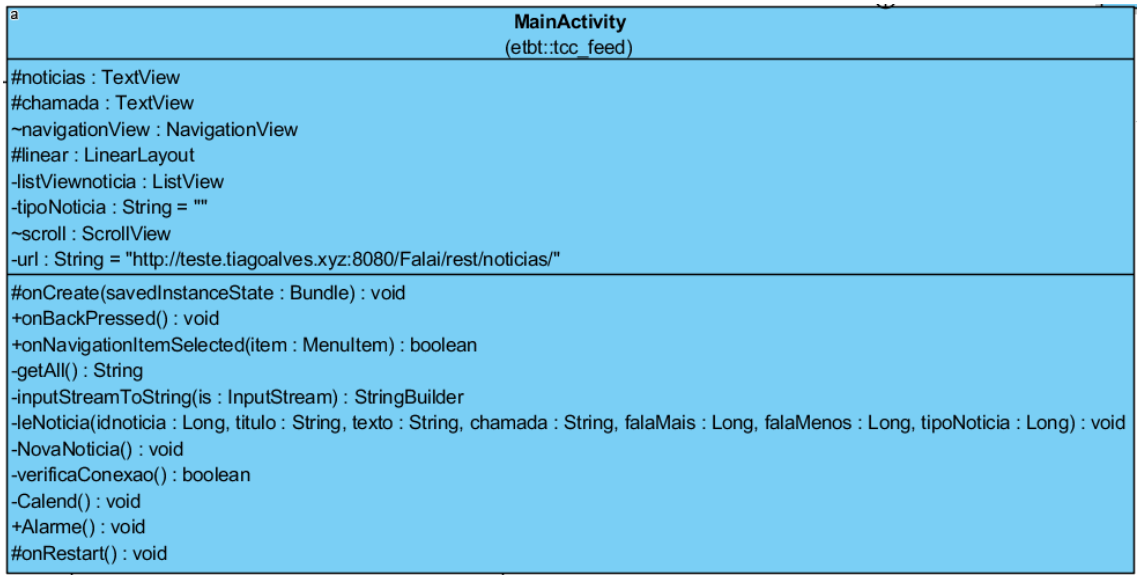
Figura 20. Esta figura representa a as classes de opções para os eventos do calendário.

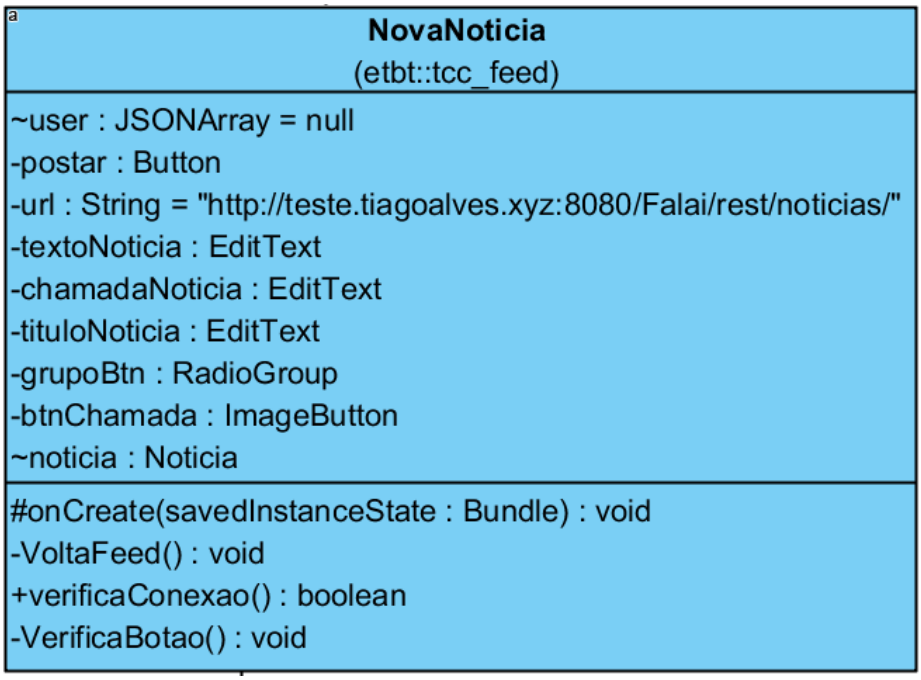
6.1. Classes

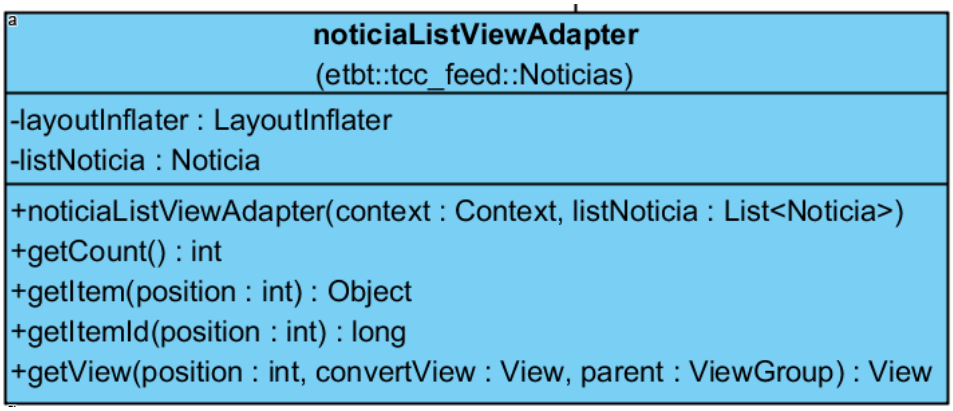
Agora será apresentado, cada classe com seu respectivos métodos e atributos.

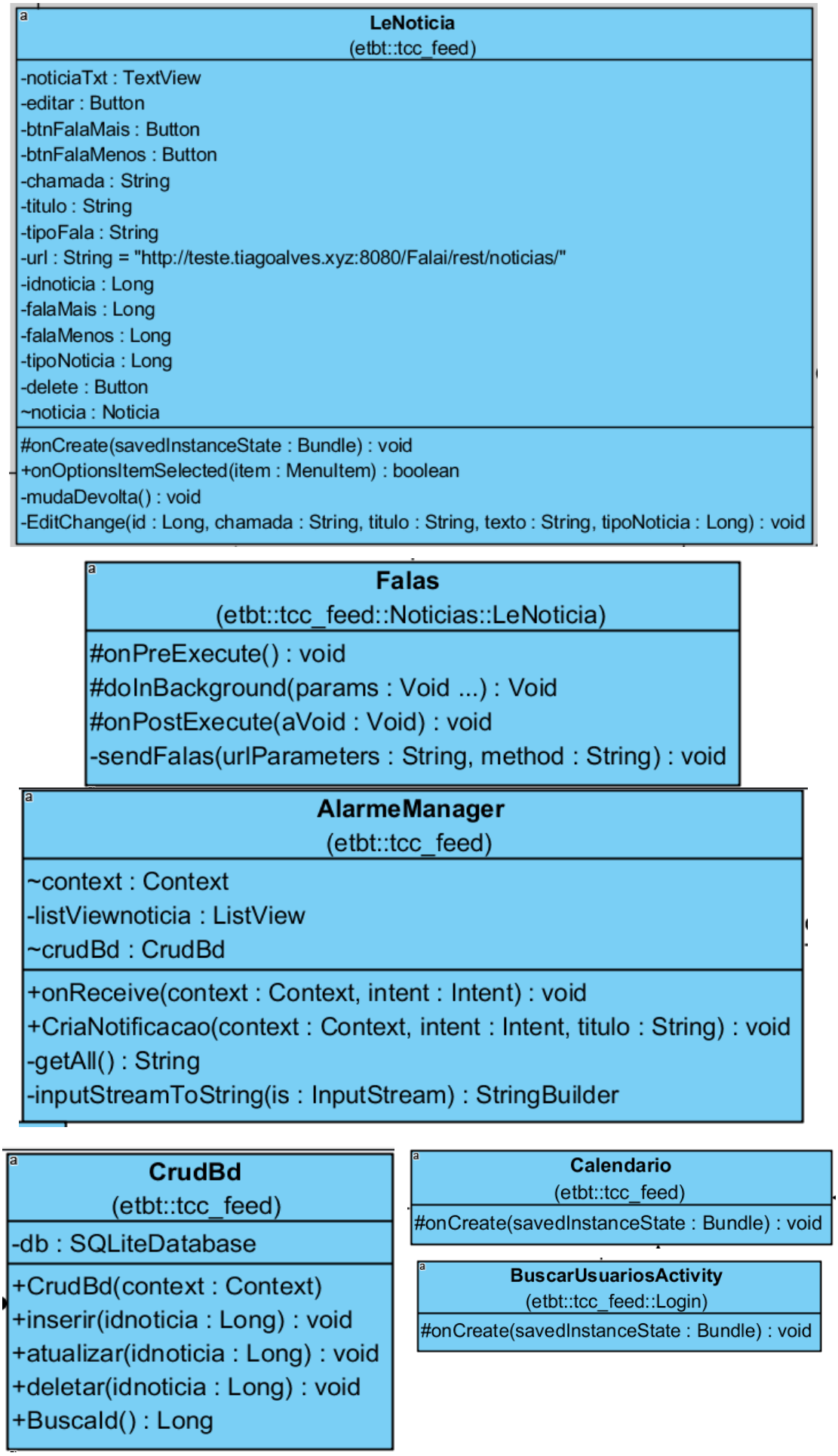


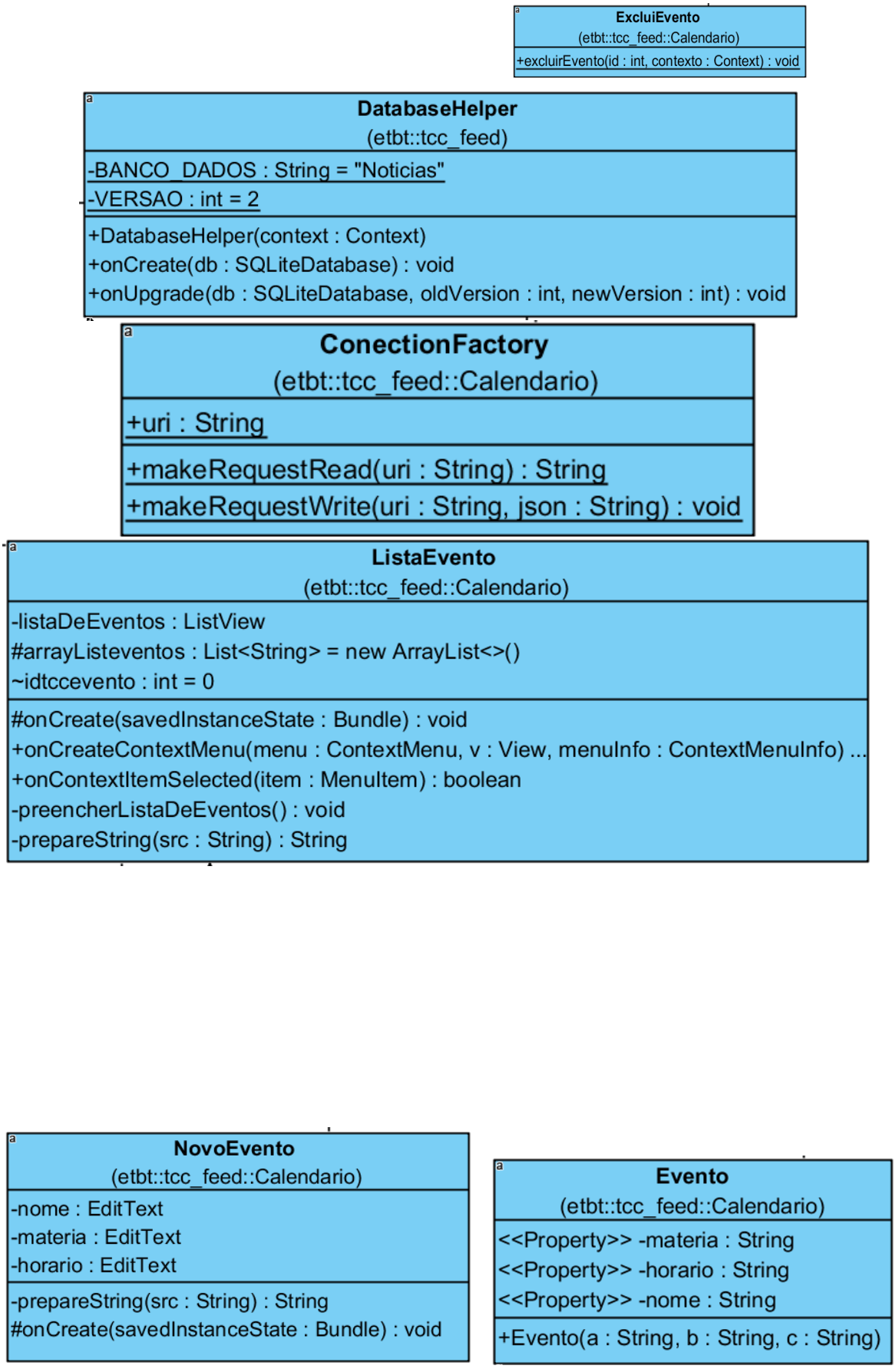

HttpAsyncTask
(etbt::tcc_feed::MainActivity)
#doInBackground(URLs : String ...) : String
#onPostExecute(result : String) : void
#doInBackground(URLs : String ...) : String
#onPostExecute(result : String) : void
DatabaseHelperLogin
(etbt::tcc_feed::SQLite)
-BANCO_DADOS : String = "BancoLogin"
-VERSAO : int = 1
+DatabaseHelperLogin(context : Context)
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
UsuarioDAO
(etbt::tcc_feed::Login)
-URL : String = "http://192.168.56.1:8080/ExemploWS/services/UsuarioDAO?wsdl"
-NAMESPACE : String = "http://login.tcc.br.com"
-INSERIR : String = "inserirUsuario"
-EXCLUIR : String = "atualizarUsuario"
-ATUALIZAR : String = "excluirUsuario"
-BUSCAR_TODOS : String = "buscarTodosUsuarios"
-BUSCAR_POR_EMAIL : String = "buscarUsuarioPorID"
+inserirUsuario(usuario : Usuario) : boolean
+atualizarUsuario(usuario : Usuario) : boolean
+excluirUsuario(usuario : Usuario) : boolean
+excluirUsuario(id : int) : boolean
+buscarUsuarioPorEmail(email : String) : Usuario
postNoticia
(etbt::tcc_feed::Noticias::NovaNoticia)
#onPreExecute() : void
#doInBackground(params : Void ...) : Void
#onPostExecute(aVoid : Void) : void
-sendPost(urlParameters : String, method : String) : void
Noticia
(etbt::tcc_feed)
-id : Long
-textoNoticia : String
<<Property>> -chamada : String
<<Property>> -titulo : String
<<Property>> -falaMais : Long
<<Property>> -tipoNoticia : Long
<<Property>> -falaMenos : Long
<<Property>> -report : Long
+getID() : Long
+setID(id : Long) : void
+getTexto() : String
+setTexto(textoNoticia : String) : void
+Noticia()
+Noticia(id : Long, textoNoticia : String, chamada : String, titulo : String, falaMais : Long, tipoNoticia : Long, falaMenos : Long, report : Long)
+toString() : String
DeleteNoticia
(etbt::tcc_feed::Noticias::LeNoticia)
#doInBackground(params : Void ...) : Void

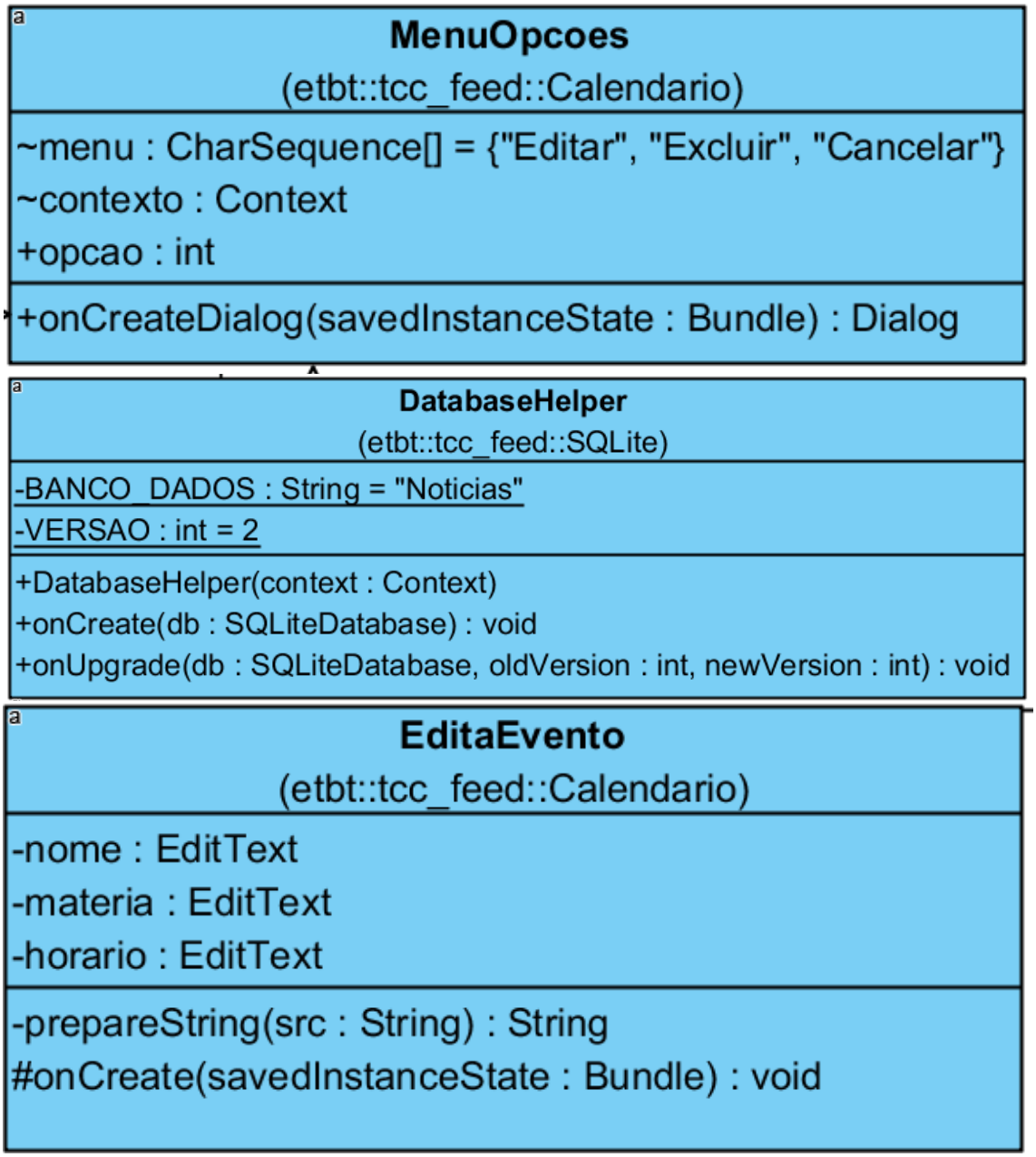

MainActivity
(etbt::tcc_feed)
#noticias : TextView
#chamada : TextView
~navigationView : NavigationView
#linear : LinearLayout
-listViewnoticia : ListView
-tipoNoticia : String = ""
~scroll : ScrollView
-url : String = "http://teste.tiagoalves.xyz:8080/Falai/rest/noticias/"
#onCreate(savedInstanceState : Bundle) : void
+onBackPressed() : void
+onNavigationItemSelected(item : MenuItem) : boolean
-getAll() : String
-inputStreamToString(is : InputStream) : StringBuilder
-leNoticia(idnoticia : Long, titulo : String, texto : String, chamada : String, falaMais : Long, falaMenos : Long, tipoNoticia : Long) : void
-NovaNoticia() : void
-verificaConexao() : boolean
-Calend() : void
+Alarme() : void
#onRestart() : void

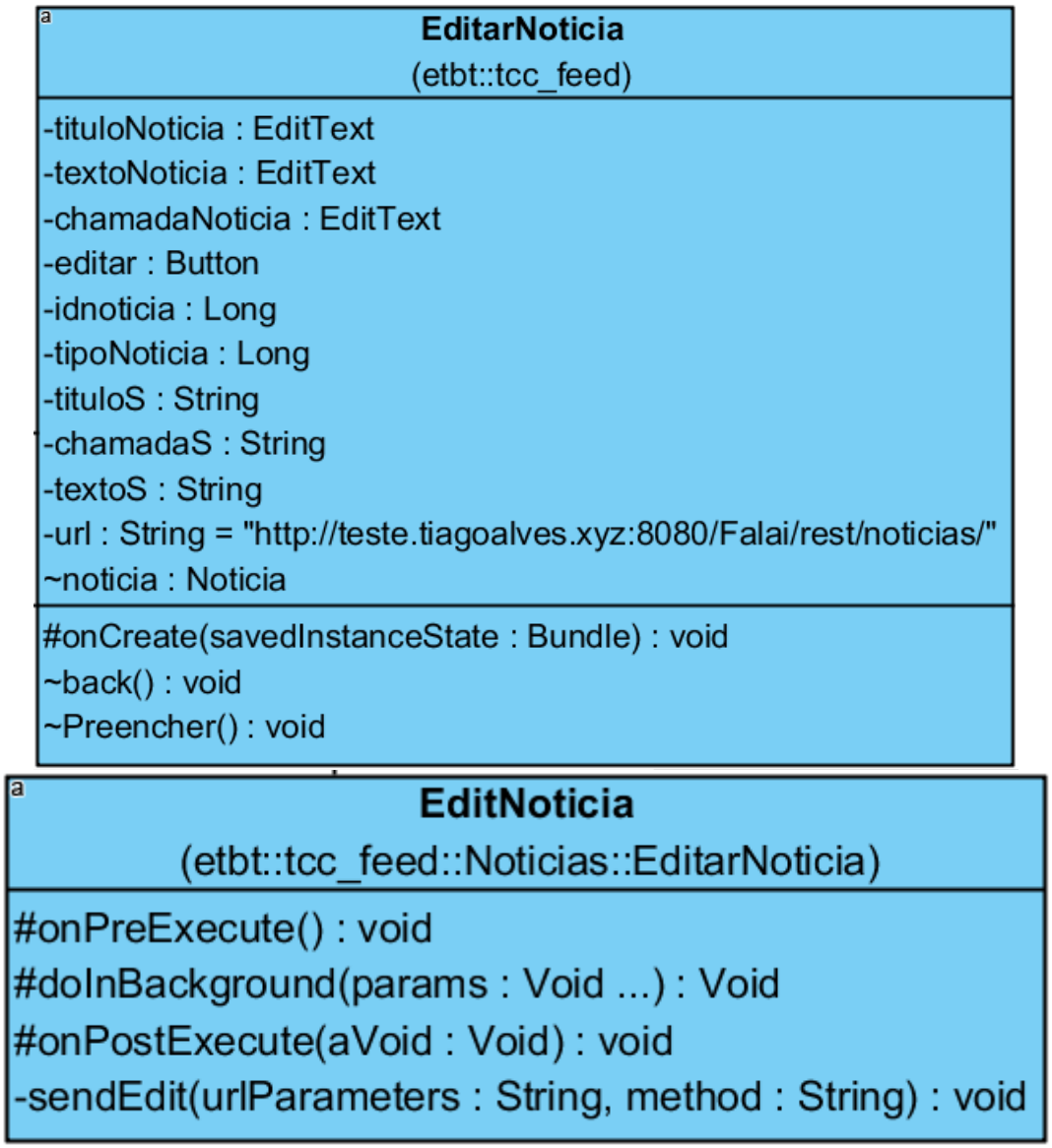

NovaNoticia
(etbt::tcc_feed)
~user : JSONArray = null
-postar : Button
-url : String = "http://teste.tiagoalves.xyz:8080/Falai/rest/noticias/"
-textoNoticia : EditText
-chamadaNoticia : EditText
-tituloNoticia : EditText
-grupoBtn : RadioGroup
-btnChamada : ImageButton
~noticia : Noticia
#onCreate(savedInstanceState : Bundle) : void
-VoltaFeed() : void
+verificaConexao() : boolean
-VerificaBotao() : void


noticiaListViewAdapter
(etbt::tcc_feed::Noticias)
-layoutInflater : LayoutInflater
-listNoticia : Noticia
+noticiaListViewAdapter(context : Context, listNoticia : List<Noticia>)
+getCount() : int
+getItem(position : int) : Object
+getItemId(position : int) : long
+getView(position : int, convertView : View, parent : ViewGroup) : View


LeNoticia
(etbt::tcc_feed)
-noticiaTxt : TextView
-editar : Button
-btnFalaMais : Button
-btnFalaMenos : Button
-chamada : String
-titulo : String
-tipoFala : String
-url : String = "http://teste.tiagoalves.xyz:8080/Falai/rest/noticias/"
-idnoticia : Long
-falaMais : Long
-falaMenos : Long
-tipoNoticia : Long
-delete : Button
~noticia : Noticia
#onCreate(savedInstanceState : Bundle) : void
+onOptionsItemSelected(item : MenuItem) : boolean
-mudaDevolta() : void
-EditChange(id : Long, chamada : String, titulo : String, texto : String, tipoNoticia : Long) : void
Falas
(etbt::tcc_feed::Noticias::LeNoticia)
#onPreExecute() : void
#doInBackground(params : Void ...) : Void
#onPostExecute(aVoid : Void) : void
-sendFalas(urlParameters : String, method : String) : void
AlarmeManager
(etbt::tcc_feed)
~context : Context
-listViewnoticia : ListView
~crudBd : CrudBd
+onReceive(context : Context, intent : Intent) : void
+CriaNotificacao(context : Context, intent : Intent, titulo : String) : void
-getAll() : String
-inputStreamToString(is : InputStream) : StringBuilder
CrudBd
(etbt::tcc_feed)
-db : SQLiteDatabase
+CrudBd(context : Context)
+inserir(idnoticia : Long) : void
+atualizar(idnoticia : Long) : void
+deletar(idnoticia : Long) : void
+BuscaId() : Long
Calendario
(etbt::tcc_feed)
#onCreate(savedInstanceState : Bundle) : void
BuscarUsuariosActivity
(etbt::tcc_feed::Login)
#onCreate(savedInstanceState : Bundle) : void


ExcluiEvento
(etbt::tcc_feed::Calendario)
+excluirEvento(id : int, contexto : Context) : void
DatabaseHelper
(etbt::tcc_feed)
-BANCO_DADOS : String = "Noticias"
-VERSAO : int = 2
+DatabaseHelper(context : Context)
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
ConectionFactory
(etbt::tcc_feed::Calendario)
+uri : String
+makeRequestRead(uri : String) : String
+makeRequestWrite(uri : String, json : String) : void
ListaEvento
(etbt::tcc_feed::Calendario)
-listaDeEventos : ListView
#arrayListeventos : List<String> = new ArrayList<>()
~idtccevento : int = 0
#onCreate(savedInstanceState : Bundle) : void
+onCreateContextMenu(menu : ContextMenu, v : View, menuInfo : ContextMenuInfo) ...
+onContextItemSelected(item : MenuItem) : boolean
-preencherListaDeEventos() : void
-prepareString(src : String) : String
NovoEvento
(etbt::tcc_feed::Calendario)
-nome : EditText
-materia : EditText
-horario : EditText
-prepareString(src : String) : String
#onCreate(savedInstanceState : Bundle) : void
Evento
(etbt::tcc_feed::Calendario)
<<Property>> -materia : String
<<Property>> -horario : String
<<Property>> -nome : String
+Evento(a : String, b : String, c : String)


MenuOpcoes
(etbt::tcc_feed::Calendario)
~menu : CharSequence[] = {"Editar", "Excluir", "Cancelar"}
~contexto : Context
+opcao : int
+onCreateDialog(savedInstanceState : Bundle) : Dialog
DatabaseHelper
(etbt::tcc_feed::SQLite)
-BANCO_DADOS : String = "Noticias"
-VERSAO : int = 2
+DatabaseHelper(context : Context)
+onCreate(db : SQLiteDatabase) : void
+onUpgrade(db : SQLiteDatabase, oldVersion : int, newVersion : int) : void
EditaEvento
(etbt::tcc_feed::Calendario)
-nome : EditText
-materia : EditText
-horario : EditText
-prepareString(src : String) : String
#onCreate(savedInstanceState : Bundle) : void


EditarNoticia
(etbt::tcc_feed)
-tituloNoticia : EditText
-textoNoticia : EditText
-chamadaNoticia : EditText
-editar : Button
-idnoticia : Long
-tipoNoticia : Long
-tituloS : String
-chamadaS : String
-textoS : String
-url : String = "http://teste.tiagoalves.xyz:8080/Falai/rest/noticias/"
~noticia : Noticia
#onCreate(savedInstanceState : Bundle) : void
~back() : void
~Preencher() : void
EditNoticia
(etbt::tcc_feed::Noticias::EditarNoticia)
#onPreExecute() : void
#doInBackground(params : Void ...) : Void
#onPostExecute(aVoid : Void) : void
-sendEdit(urlParameters : String, method : String) : void