HaQton – Competição de Programação da QtCon Brasil 2020
Regulamento

1. INTRODUÇÃO

Este documento apresenta os objetivos, formato e o regulamento geral do HaQton – competição de programação a ser realizada como parte da QtCon Brasil 2020. A QtCon Brasil é um fórum único no Brasil e na América-Latina para discussão de ideias, troca de experiências e oportunidades de negócio em torno do Qt, reunindo desenvolvedores, empresas, instituições governamentais e universidades. A QtCon Brasil 2020 contou com dois dias de palestras e oportunidades de network para profissionais interessados no uso do Qt para desenvolvimento de sistemas *desktop*, *mobile*, embarcados e de IoT. O HaQton é uma atividade adicional, organizada e promovida pela Qmob Solutions e conta com o patrocínio da openSUSE, Toradex e B2Open Systems.

2. OBJETIVOS

O HaQton tem como objetivo promover o uso do Qt na comunidade Brasileira e Latino-Americana através de uma competição de programação. Nesta competição, as equipes inscritas irão desenvolver uma aplicação Qt que requer o uso de funcionalidades relacionadas a UIs, multimídia, comunicação em rede via APIs RESTful e comunicação via barramentos de mensagens. Com isso, esperamos que novos talentos passem a integrar o ecossistema Qt brasileiro e latino-americano, através de discussões no canal Qt Brasil no Telegram, participação como prováveis palestrantes em edições futuras da QtCon Brasil, colaborações em projetos tais como o KDE, etc.

3. REGULAMENTO

As seções a seguir descrevem o regulamento do HaQton. A Seção 3.1 apresenta o sistema a ser desenvolvido, discutindo os objetivos gerais da aplicação e os seus requisitos funcionais e não-funcionais. A Seção 3.2 informa como realizar as inscrições e as condições de elegibilidade das equipes. A Seção 3.3 detalha os critérios de avaliação e a Seção 3.4 informa o prazo para entrega final do sistema desenvolvido. Finalmente, a Seção 3.5 informa a premiação definida para a competição.

3.1. SISTEMA A SER DESENVOLVIDO

Equipes inscritas nesta competição terão como tarefa o desenvolvimento de um jogo *multiplayer* e on-line do tipo *quiz*. Neste jogo, múltiplos jogadores participarão de uma determinada partida e tentarão, um de cada vez, responder corretamente uma determinada pergunta exibida na aplicação. As perguntas serão sempre do tipo múltipla escolha, abordando assuntos diversos e com apenas uma resposta correta.

Para permitir que o jogo funcione em modo *multiplayer* e on-line, a aplicação precisará se comunicar com dois servidores: um *back-end* RESTful (para gerenciamento das informações sobre partidas, participantes, pontuação e perguntas – https://haqton.qmob.solutions); e um barramento de mensagens (para sincronização das jogadas e das partidas) – tcp://zmq.qmob.solutions:5560. Estes dois servidores foram desenvolvidos e disponibilizados pela Qmob Solutions e não precisarão ser implementados pelas equipes. O *back-end* RESTful foi desenvolvido utilizando a linguagem de programação Ruby e o *framework* Sinatra. Cada equipe inscrita na competição receberá um token JWT para uso na comunicação com o *back-end* RESTful. O barramento de mensagens foi desenvolvido utilizando o ZeroMQ. A documentação de API do *back-end* RESTful, com indicação das mensagens ZeroMQ enviadas a cada momento, está disponível em http://bit.do/api-haqton.

Todas as equipes irão trabalhar a partir de um esqueleto inicial do sistema, disponibilizado pela Qmob Solutions em https://gitlab.com/qmob-solutions-projects/haqton/haqton-client. Este esqueleto contém funcionalidades mínimas para autenticação, envio e recepção de mensagens e uma versão preliminar do menu inicial da aplicação. Todas as demais funcionalidades, descritas a

seguir, deverão ser implementadas pelas equipes. Cada equipe deverá fazer um *fork* deste repositório e informar a URL aos organizadores da competição.

Tela inicial da aplicação

Requisitos funcionais:

- A primeira tela da aplicação deve solicitar o nickname e o e-mail do usuário.

RF1: Criação de nova partida

- A partir do menu principal, selecionar o botão "Criar nova partida".
- Solicitar que o usuário digite o nome da partida.
- Realização uma invocação *POST Create Match* (ver documentação em http://bit.do/api-haqton) para criar a nova partida.
- Navegar (push na stackView) para a tela de aprovação de participantes (RF3).

- Utilizar MessagingController::setTopic() para passar a ouvir mensagens do tópico associado à partida recém-criada.

RF2: Visualização das partidas aguardando jogadores

- A partir do menu principal, selecionar o botão "Participar de uma partida".
- Realizar uma invocação *GET Matches Waiting for Players* (ver documentação em http://bit.do/api-haqton) para obter a lista de partidas aguardando jogadores e exibir as informações obtidas.
- No recebimento de uma mensagem ZeroMQ do tipo "*matches_update*" atualizar a lista de partidas.
- Ao clicar em uma determinada partida, realizar uma invocação *POST Add Player to Match* (ver documentação em http://bit.do/api-haqton). Esta mesma invocação já retornará a lista atual de participantes da partida.
- Navegar (push na stackView) para a tela de aprovação de participantes (RF4).

RF3: Tela de aprovação de participantes (visão do criador da partida)

- Apresentar, na tela da aplicação, a mensagem "Aguardando chegada dos jogadores".
- No recebimento de uma mensagem ZeroMQ do tipo "*players_update*" atualizar a lista de participantes da partida.
- Para cada participante, devem ser apresentados dois botões: um de aprovação de participação e outro de reprovação de participação.
    - Ao clicar no botão de aprovação ou reprovação, realizar uma invocação *PUT Update Player* (ver documentação em http://bit.do/api-haqton), informando o novo status (propriedade *approved*) de participação do usuário (concedido ou negado).
- No canto inferior da tela deve ser apresentado um botão "Iniciar Partida" que, ao clicado, fará a navegação (push na stackView) para a tela de rodada do jogo (RF5). Neste momento, deve ser realizada uma invocação *GET Random Question* (ver documentação em http://bit.do/api-haqton) para obter a primeira pergunta do jogo.
- O botão "Iniciar Partida" só deverá ser habilitado após o criador indicar, para cada participante da partida, se o seu acesso será concedido ou negado.

RF4: Tela de aprovação de participantes (visão do participante)

- Para cada participante da partida, devem ser apresentados dois ícones: um de "participação aprovada" e outro de "aguardando aprovação".
- No recebimento de uma mensagem ZeroMQ do tipo "*players_update*" atualizar a lista de participantes da partida. Esta mensagem é enviada quando novos jogadores solicitam a participação em uma partida ou quando o criador da partida concede ou rejeita alguma participação.
- No recebimento de uma mensagem ZeroMQ do tipo "*new_question*", navegar (push na stackView) para a tela de rodada de jogo (RF5).

RF5: Tela de rodada do jogo (visão do criador e do participante)

- Apresentar os dados da pergunta (obtidas na mensagem do tipo "*new_question*"). Os dados da pergunta contêm: o enunciado da questão, todas as respostas possíveis e o índice da resposta correta.
- Abaixo dos dados da pergunta, apresentar, para cada participante, um ícone sinalizando o status do participante naquela rodada ("participante ainda não respondeu" ou "participante já respondeu") e o seu score (cada resposta correta vale 1 ponto).
- No recebimento de uma mensagem ZeroMQ do tipo "*new_answer*" atualizar o ícone de sinalização do participante indicado na mensagem de "participante ainda não respondeu" para "participante já respondeu" e atualizar o score do participante que respondeu na UI.
- Ao selecionar uma resposta para a pergunta, deverá ser realizada uma invocação *POST Process Player Answer* (ver documentação em http://bit.do/api-haqton).
- Um botão "Encerrar Partida" deve estar presente e visível apenas para os criadores de uma partida. Ao ser clicado, uma invocação *PUT Update Match* (ver documentação em http://bit.do/api-haqton) deve ser realizada, modificando o status da partida para 2 (encerrada).
- O criador da partida (e somente ele) deve controlar o número de respostas (mensagens do tipo "*new_answer*") recebidas. Quando este número de respostas for igual ao número de

participantes da partida as respostas de cada jogador devem ser reveladas e deverá ser realizada uma nova invocação *GET Random Question* (ver documentação em http://bit.do/api-haqton) para obter a próxima pergunta do jogo. Um botão "Próxima Pergunta" deve ser apresentado. Ao ser clicado, a tela é atualizada com a próxima pergunta obtida e o processo se repete.

- No recebimento de uma mensagem ZeroMQ do tipo "*match_finished*", apresentar uma mensagem de fim de partida e um botão "Ver Resultado Final". Ao ser clicado, a tela de fim de partida é apresentada. A mensagem "*match_finished*" é emitida ou quando o número de questões se esgotou para aquela partida (seu estado é automaticamente modificado para 2 – encerrada) ou quando o criador explicitamente encerra a partida (através do botão "Encerrar Partida").
- **OBS:** o controle dos pontos deve ser feita exclusivamente no lado do front-end. O endpoint *POST Process Player Answer* já verifica se a resposta fornecida é correta ou não e atualiza o score do jogador corretamente.

RF6: Tela de fim de partida

- Esta tela deve apresentar os jogadores da partida e os seus respectivos pontos, em ordem descrescente de pontuação.
- No canto inferior da tela deve ser apresentado um botão "Retornar" que, ao clicado, fará a navegação (pop na stackView) para a tela inicial da aplicação.

Requisitos não-funcionais:

RNF1: Desempenho

As atualizações das telas, como consequência do recebimento de mensagens, devem ser imediatas e não apresentar atrasar desnecessários. O ZeroMQ possui desempenho satisfatório o suficiente para que não ocorram atrasos consideráveis de rede.

RNF2: Usabilidade

Todas as UIs da aplicação devem ser desenvolvidas com foco em boa usabilidade, permitindo a realização das operações com um número reduzido de clicks e apresentando um *look'n'feel* compatível e adequado para um jogo.

RNF3: Manutenibilidade

A arquitetura e o código-fonte da aplicação devem ser projetados e implementados de modo a partir a fácil manutenção e evolução da aplicação. O esqueleto fornecido pela Qmob Solutions já disponibiliza uma implementação inicial do padrão arquitetural *Microkernel*, com a presença de um componente *Core* e de um *MessagingController*. É fortemente recomendado que as equipes sigam este padrão arquitetural, adicionando novos *controllers* conforme necessário. Soluções apresentando código-fonte com baixa legibilidade, arquiteturas altamente acopladas ou algoritmos demasiadadmente complexos serão penalizadas.

RNF4: Suporte a múltiplas plataformas

A competição é sobre desenvolvimento com Qt e, portanto, não podemos deixar de nos beneficiar de uma das maiores vantagens do Qt: o suporte a múltiplas plataformas. A solução apresentada pelas equipes deve funcionar em pelo menos uma plataforma desktop (Linux, Windows ou macOS) e pelo menos uma plataforma mobile (Android ou iOS). No momento da entrega, a equipe indicará em quais plataformas a solução foi testada. Soluções contemplando plataformas adicionais (embarcadas, web, etc) serão adicionalmente bonificadas.

NOTA: todos os artefatos desenvolvidos pela equipe vencedora serão livremente disponibilizados sob a licença GPLv3 no repositório GitLab da Qmob Solutions. Mais informações sobre os formatos das mensagens, formato das invocações RESTful e repositório do esqueleto inicial da aplicação serão enviadas após a homologação das equipes inscritas.

3.2. SOBRE AS INSCRIÇÕES

As equipes podem ser formadas por 1 ou 2 participantes e as inscrições devem ser realizadas no link http://bit.do/haqton entre os dias 29 de setembro e 04 de outubro. Todas as inscrições serão manualmente homologadas e a equipe será notificada da aceitação/rejeição por e-mail.

Todos os membros da equipe devem atender às seguintes condições de elegibilidade:

- Não ter mais de 2 anos de experiência com Qt.
- Não possui parentesco de primeiro grau com colaboradores da Qmob Solutions, openSUSE, Toradex e B2Open Systems.
- Estar inscrito como participante da QtCon Brasil 2020.
- Ter 18 anos ou mais no momento da inscrição no HaQton.

3.3. CRITÉRIOS DE AVALIAÇÃO

Os trabalhos desenvolvidos pelas equipes serão avaliados de acordo com o seguinte barema:

BAREMA DE AVALIAÇÃO					
Critério	**Não atendido**	**Insatisfatoriamente atendido**	**Satisfatoriamente atendido**	**Extra mile**	**Peso**
RF1	-1	1	3	5	3
RF2	-1	1	3	5	5
RF3	-1	1	3	5	10
RF4	-1	1	3	5	10
RF5	-1	1	3	5	30
RF6	-1	1	3	5	2
RNF1	-1	1	3	5	5
RNF2	-1	1	3	5	13

RNF3	-1	1	3	5	12
RNF4	-1	1	3	5	5
Prazo	3 pontos (-0.5 por dia de atraso, limitado a 3 dias)				5

Os trabalhos serão avaliados por uma comissão independente e experiente em projetos Qt e não terá, em sua formação, membros ligados à Qmob Solutions, openSUSE, Toradex ou B2Open Systems.

Máximo possível de pontos: 490 pontos
Número mínimo de pontos para premiação: 225 pontos

3.4. CRONOGRAMA

Inscrições das equipes	29/09 a 04/10
Homologação das inscrições	Até 06/10
Início da competição	07/10
Data final de entrega dos artefatos	18/11 às 23h59min

3.5. PREMIAÇÃO

Esta competição admitirá um máximo de duas equipes vencedoras, cuja premiação é definida a seguir.

1º lugar:

- Prêmio em dinheiro no valor de R$ 10.000,00 para a equipe.
- 1 kit de desenvolvimento embarcado Toradex (Colibri i.MX7 Dual 1GB + Aster Carrier Board) para cada integrante da equipe.
- 4h de consultoria em desenvolvimento embarcado com Qt e Yocto para a equipe.

2º lugar:

- Prêmio em dinheiro no valor de R$ 5.000,00 para a equipe.

- 1 kit de desenvolvimento embarcado Toradex (Colibri i.MX7 Dual 1GB + Aster Carrier Board) para cada integrante da equipe.
- 4h de consultoria em desenvolvimento embarcado com Qt e Yocto para a equipe.

Voltamos a ressaltar que equipes que não atingirem o número mínimo de pontos (225) não serão premiadas.

O HaQton da QtCon Brasil 2020 é gentilmente patrocinado pela openSUSE, Toradex e B2Open Systems.